Grid Domains for Analysing Software

by

Katy Louise Dobson

**Submitted in accordance with the requirements
for the degree of Doctor of Philosophy.**

**The University of Leeds
School of Computing**

August 2008

Declarations

Some parts of the work presented in Chapters 3, 4 and 5 have been published in the following article:

Bagnara, R. and Dobson, K. and Hill, P. M. and Mundell, M. and Zaffanella, E., "Grids: A Domain for Analyzing the Distribution of Numerical Values", *Logic-based Program Synthesis and Transformation, 16th International Symposium*, Lecture Notes in Computer Science, Volume 4407 (2007)

The work of this publication was joint work by all authors and it is not possible to distinguish one authors contribution from the next. A copy of this work is given in the appendix.

Acknowledgements

My supervisor Dr. Patricia M. Hill deserves a huge amount of praise for her unfaltering patience, belief and encouragement throughout my research. Pat has taught me so much, she has been instrumental in my research and without Pat I would not have been introduced to Prof. Roberto Bagnara and Prof. Enea Zaffanella or the wonderful world of computer science. Roberto and Enea have welcomed me into their team and allowed me to work on this amazing project, their help and knowledge has been immeasurable to me.

I would like to thank all of the people with whom I have been able to discuss my research with and who have provided invaluable feedback, notably Dr. Javier Núñez-Fontarnau, Vajirapan Panumong, Matthew Mundell, Prof. Fausto Spoto and Dr. Andy King. I would also like to acknowledge Dr. Philippe Granger for providing me with a copy of his thesis. His work has been an inspiration. I wish to thank The School of Computing, University of Leeds and the Frank Parkinson Scholarship for funding my research.

Finally I would like to thank my family as this work would not have been possible without them, they have made me all that I am. Without my grandmother Betty I would not have the tenacity and courage to finish this work, without my brother Stuart I would not have the imagination and expertise to start this work and without my mother Susan I would not have the strength, passion and integrity to do anything. My words can not express what you all mean to me, I dedicate this work to you. I also dedicate this work to my father Tony and my grandfather Albert. You were both taken too soon and I miss you both.... too much

Abstract

Static analysis is the determination of correct though approximate information about the behaviour of a system, this approach is used to detect and locate programming errors or to certify the absence of such bugs. Abstract interpretation is a static program analysis method that uses *abstract domains* to provide a convenient but approximate representation of the accumulated information during the evaluation of a program. The focus of this thesis is to investigate numerical abstract domains that capture the distribution or patterns of values the program properties can take. There has already been a considerable amount of research into numerical abstract domains and a wide variety of such domains have been specified each providing a different degree of precision and efficiency. For instance the domain of convex polyhedra is precise but has exponential complexity while the interval or box domain is much less precise but has linear complexity. Note that these domains do not capture the distribution information which is the focus of this thesis.

In the first part of this thesis we introduce the domain of *grids*. This domain interprets the patterns of distribution of the values that the program properties can take. The complete grid domain can interpret the relationships which hold between variables or properties in a program. There are two representations that form the two components of a double description method similar to that provided for convex polyhedra. This thesis gives algorithms and methods for computing canonical forms, conversion between the descriptions and the main abstract operations needed for software analysis, such as comparison, intersection, join, difference, affine image and pre-image. Also included is a widening operation and we show that all of these operations have polynomial complexity.

In the second part of this thesis we consider the *partially reduced product* of two numerical domains. The partially reduced product allows a choice of interaction between the component domains ranging from "do nothing" required by the direct product to a total reduction required by the reduced product. We consider the partially reduced product where the components are those of the grid domain with either the convex polyhedra domain or one of its sub-domains, specifically the boxes, bounded difference shapes and octagon domains. The "weakly tight product" is introduced, an operation that ensures each constraint of the polyhedral representation intersects a point of the grid, and the "tight product", which ensures each constraint of the polyhedral representation intersects a point of the grid-polyhedron. We provide an algorithm to compute the weakly tight product and show for what circumstances this algorithm achieves stronger results, so that the resulting grid-polyhedron is either a tight or a reduced product. Methods for testing if a grid-polyhedron is empty as well as several useful operations on grid-polyhedra are also described.

Contents

List of Figures

List of Tables

Try not. Do or do not. There is no try.
Yoda
Star Wars: The Empire Strikes Back

All of your dreams can come true if you have the courage to pursue them.
Walt Disney

Chapter 1

The Introduction

It is widely known that computers are everywhere, they are used in almost every aspect of everyday life, from controlling power stations that produce our energy to controlling the bank accounts that contain the world's money. Therefore it has never been more important to know that the software these computers run is safe and also efficient, however it would also be beneficial if the method for testing the safety and efficiency were also accurate and efficient. The costs of software errors are not only monetary, they can also have an impact on everyday life. In 1999 the Mars Climate Orbiter was lost on entering the Mars atmosphere at a cost of $328 million. The failure investigation team found that "a lack of complete end-to-end verification of navigation software and related computer models" was a key factor in the failure, see `http://mars.jpl.nasa.gov/msp98/news/mco991110.html`. Also in August 2003 an unknown software flaw caused a blackout in parts of Canada and the northeastern United States. The flaw in a widely-deployed General Electric energy management system contributed to the devastating scope of the blackout. The bug in GE Energy's XA/21 system was discovered in an intensive code audit conducted by GE. "It had never evidenced itself until that day," said spokesman Ralph DiNicola, "this fault was so deeply embedded, it took them weeks of poring through millions of lines of code and data to find it," see `http://www.securityfocus.com/news/8016`. The cascading blackout eventually cut off electricity to 50 million people in parts of Canada and eight states of America.

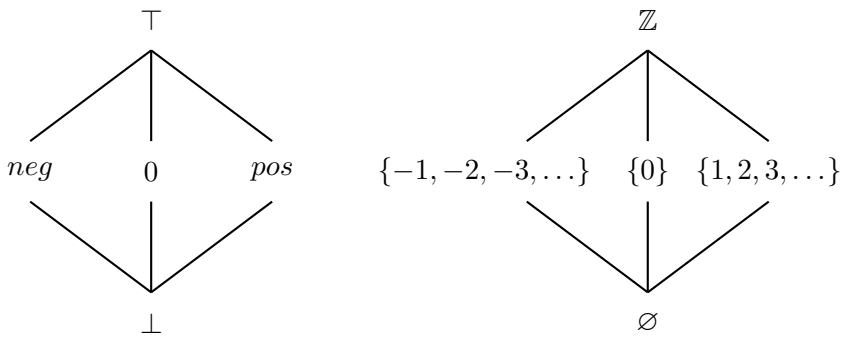
We are interested in looking at program analysis of which there are two main ways to analyse program properties, dynamically and statically. Dynamic analysis executes the program code and uses large sets of data as inputs to see if any interesting behaviour occurs, unfortunately it is this choice of input that can dramatically alter the output. Therefore we are concerned with static analysis which doesn't actually execute the program code, it instead approximates the behaviour.

The most precise way to analyse a piece of code is to consider the exact (also called concrete) semantics, however these are often very complicated. Therefore we have to decide between the precision of the analysis and the computational complexity. This is why instead of considering the exact semantics we consider an abstract semantics.

The abstract semantics are decided by the method of abstract interpretation which was introduced by Patrick and Radhia Cousot [27]. Abstract interpretation involves approximating the computations of the program by new computations over an abstract domain which is known to be simpler. Then when the abstract computations are performed it is hoped that the information yielded from the abstract domain will shed light on the possible results the actual computations would have provided. The soundness of this process is ensured by a pair of mappings between the concrete and abstract semantics, these mappings show how the elements of one domain should be interpreted in terms of the other.

Informally, consider the following example. Suppose a police force wish to search for a suspect in a database of every convicted criminal in the UK. The exact (or concrete) way to find the suspect is to look for a person with that name, date of birth, last known address and national insurance number (if it is known). By considering all of these criteria and other possible distinguishing characteristics the police should be guaranteed to find the correct suspect. Alternatively, an abstract approach would be to approximate the suspects criteria, for example we could just look at all the people with the same name. Note that this would create a quicker search as less deciding factors have to be met but this method will gather a possible set of results which would contain the suspect but also possibly give extra people. These extra results are called *false alarms*.

There are several well researched abstract domains, each tailored to the type of information they wish to investigate and analyse. In this thesis we will be concerned with numerical abstract domains which consider linear information. We can classify the the types of numerical information into two groups: the *limits* or bounds within which the values can take and the *distribution* of the values to see if any pattern occurs. The study of both types of numerical information have their applications. Applications which require the distribution of values to be observed include data dependence analysis for arrays which are required for advanced optimizing compilers [70], estimating the worst case execution time of a program [19], to aid in the construction of program transformations for saving energy on low-power architectures and improving performance on multimedia processors [47] or to gather information about non-linear operations within the program [46]. Therefore the choice of abstract domain is important as it must consider the correct type of numerical data for the problem at hand. One of the simplest domains is the “rule of signs”, where the integer values are abstracted to pos, neg or 0 depending on whether they are positive integers, negative integers, or zero respectively [29]. The domain of signs can be represented by a lattice and can be seen in Figure 1.1(a), the corresponding concrete values over the set of integers can be seen in Figure 1.1(b). This domain can ascertain such properties as “is the variable negative at a certain point in the program”, however the domain is not sophisticated enough to establish “is the variable less than 10 at a certain point in the program”. Hence this domain was



(a) The signs domain. (b) Concrete values over $\mathbb{Z}$.

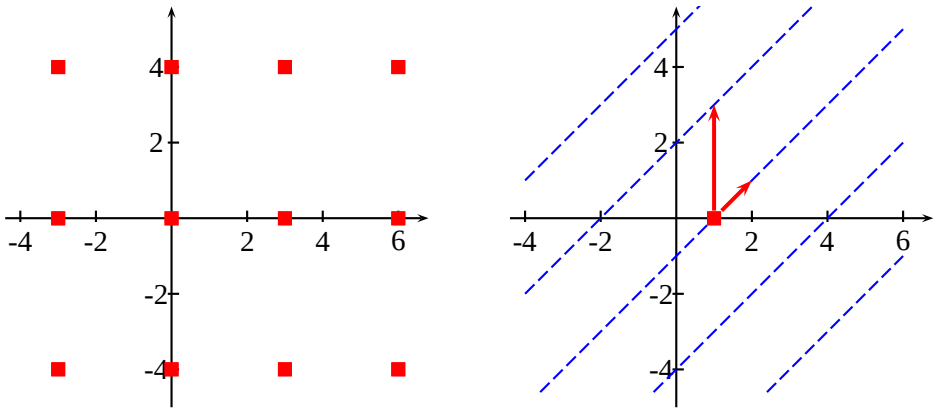
Figure 1.1: The abstract domain of signs represented as a lattice.

then generalised to the interval domain [26] which considers limit information by constructing integer upper and lower bounds for each variable, nevertheless this domain could also be improved to create more accurate information since it does not show any dependence between the variables. Hence more complex domains are required.

Often the numerical information, whether it is limit or distribution, comes in a *relational* form, that is, the values of one variable may be dependent on the values of one or more other variables. One domain that captures the linear relational limit information is the *polyhedron domain*, this domain represents regions of some n-dimensional vector space bounded by a finite set of hyperplanes [32]. There are also several different polyhedron sub-domains such as the domain of convex polyhedra [6, 7, 9, 14], octagons [4, 15, 50, 54], octahedrons [21], bounded difference shapes (bds) [49, 51, 53], two variables per inequality (tvpi) [77] and intervals [26]. Each of these abstract domains can be described by different classes of constraint as seen in Table 2.1 on Page 16 and each of these domains are illustrated by n-dimensional shapes, see Figure 2.2 on Page 17, so it can be seen that they do not capture any distribution information. Although the polyhedron domain and its sub-domains have been thoroughly researched and are widely used, relational domains for representing the (linear) distribution of numerical values have been less well researched.

1.1 The Grid Domain

This thesis considers two related topics. In the first topic of this thesis we will introduce a relational domain called the grid domain. This domain encodes information about the distribution of numerical values. The grid domain is based on the domain of congruences described by Granger [37–39, 41]. The grid domain can be used for any of the applications mentioned above and details are given in Chapter 5. In the 1-dimensional case, where the grid will define a subset of points along the line $-\infty \leq x \leq \infty$, the grid can be a single point, such as, $x = 1$, a discrete



(a) The grid $\mathcal{L}$. (b) The grid $\mathcal{L}$.

Figure 1.2: Grids in $\mathbb{R}^2$.

set of equally spaced points, for example the set of even integers, or all the points along the line. Let us now consider the 2-dimensional case. Then the grid can take many forms. The grid can be a single point, such as, $\{x = 0, y = 1\}$, a set of equally spaced points along a line, for example, the set $\{x = 0, y = 2k + 1 | k \in \mathbb{Z}\}$, a set of equally spaced points that cover a plane, for example, see Figure 1.2(a), a set of lines, for example, see Figure 1.2(b), or the whole vectorspace itself, for example $\mathbb{R}^2$. In Figure 1.2(a) the grid is given by the set of points illustrated by the squares. It can be seen that the grid is non-relational as the points lie parallel to each of the axes. In Figure 1.2(b) the grid is given by the all points along the diagonal lines. These are all the points that satisfy the congruence $x - y = 1 \mod 3$, so the grid is the set of points (x, y) such that $\{x - y = a \big| a \in \{\ldots, -2, 1, 4, \ldots\}\}$. It can be seen that the grid is not non-relational as the congruence that describes the grid involves both x and y, visually this can be seen in Figure 1.2(b) as the lines of the grid do not lie parallel to either of the axes.

Let us consider a simple example to show how the grid domain can be used to interpret a small piece of code. Figure 1.2(b) illustrates two ways of describing a grid; either by means of a finite set of congruence relations that all grid points must satisfy (given by dashed lines) or by means of a finite set of generating vectors used for constructing the grid points and lines (given by filled squares and thick lines). Consider first the following program fragment for any value of `m`:

```
for i := 1 to m
  if ... then
    x := y + 1
  else
    y := y + 3
  endif
endfor
```

The dashed lines in Figure 1.2(b) illustrates the grid $\mathcal{L}$ and marks the vectors of values of the

real variables x and y after the assignments `x := y + 1` and `y := y + 3`, assuming that nothing is known about the value of x or y. The set $\mathcal{C} = \{x - y = 1 \mod 3\}$ is also called a congruence system and describes $\mathcal{L}$. Observe that the grid $\mathcal{L}$ consists of all points that can be obtained as $\lambda\boldsymbol{\ell} + \mu\mathbf{q} + \mathbf{p}$, for any $\lambda \in \mathbb{R}$ and $\mu \in \mathbb{Z}$, where $\boldsymbol{\ell} = \binom{1}{1}, \mathbf{q} = \binom{0}{3}$ and $\mathbf{p} = \binom{1}{0}$; the vector $\boldsymbol{\ell}$, called a *line*, defines a gradient, the vector $\mathbf{q}$, called a *parameter*, defines the distance to the next line and the vector $\mathbf{p}$ is a generating point marking a position for the line (illustrated in Figure 1.2(b) by the thick diagonal and vertical line and the filled square, respectively).

We will give details of how to interpret the domain and give a complete set of abstract operations *all* of which have efficiencies better than or equal to previous proposals [38, 61]. These operations are abstract forms of the set-theoretic operations such as comparison, join, meet and difference. The advantage of a domain like the grids is that, unlike the domain of convex polyhedra, all the abstract operations will have a complexity that is polynomially bounded by the number of variables. As the domain of convex polyhedra can have operations which have unbounded or exponential complexity, the cost of performing operations can grow rapidly. Whereas, the grid domain has operations that have bounded polynomial complexity like those for the interval domain, bounded difference shapes or octagon domain. The grid domain can also express relational properties over more than two variables which the interval, bounded difference shape or octagon domain can not. We will show that aspects of the grid domain parallel those of the domain of convex polyhedra, in that, not only do both domains share the same amount of expressivity, but also both have two different representations that form a double description. In Chapter 4 we will show that we can utilise this double description by designing the abstract operations to use the representation which achieves the best complexity. As we have two descriptions we will introduce a method of conversion between the two and a minimisation algorithm which puts the representation into a minimal form suitable for an easier conversion. We will show that our algorithms for producing this minimal representation and conversion have complexities more efficient than previous proposals [38, 61]. We will also be the first to give a complete set of abstract operations as previous proposals have either not given an abstract operator, such as the difference operation [38, 39], or not given one which returns a single element of the domain, such as the join and difference operations [71, 72]. To guarantee the termination of an analysis it is often useful to have a widening operation. This operation approximates the fixpoint of a sequence. Another of the contributions of this thesis will be to introduce a widening for each of the grid descriptions as previous proposals have only considered a widening for use on the generator system [38, 41]. We will also introduce other approximations, that are not widenings, but can be used to accelerate fixpoint convergence process.

1.2 Product Domains

Very little research has been done on the combination of domains which represent the limit and distribution information, especially those which consider information which takes a relational or

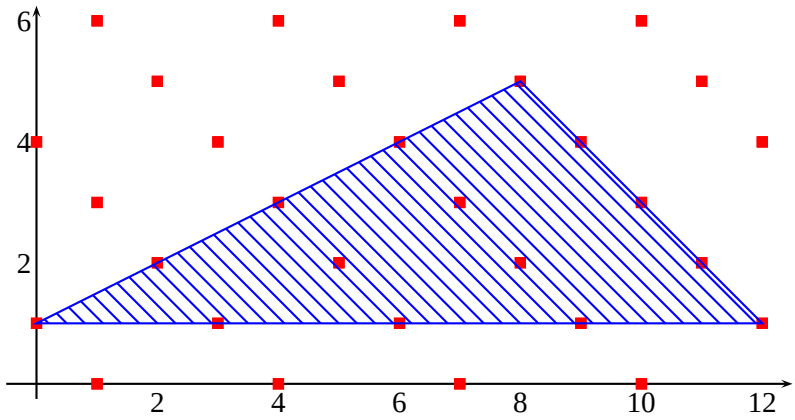


Figure 1.3: A grid-polyhedron in $\mathbb{R}^2$.

weakly relational form. The *direct* product and *reduced* product were both introduced by Cousot and Cousot [28] and take the elements of the product to be the intersection of the two components. The direct product provides a "do nothing" approach, so that there is no interaction between the components, whereas the reduced product provides a "total reduction" approach. The reduced product provides a reduction operation that ensures every element of the product is in canonical form with respect to each of the components. There are several issues we need to consider with a product domain, one is that the representation of an intersection as a pair is not canonical. Also the precision of the operations needed for an analysis is affected not only by the choice of component domains but also the allowed interactions between them, both before and during an operation. If there is no interaction then the precision gained may be very little, if any, however if there is a large amount we may greatly improve the precision but lose efficiency. Therefore the second topic we consider in this thesis is an extension of the work on the grid domain and takes a product of the grid domain with the domain of convex polyhedra or one of its simpler sub-domains. It has been shown that the grid and the many polyhedra sub-domains are useful tools for program analysis by themselves. We will introduce the *partially reduced product*. This product will allow an amount of interaction between the components, thus utilising the strengths of each domain and potentially improving the precision of the information provided compared to the results obtained by preforming the analysis separately. The partially reduced product of two numerical domains combines the direct product with several different reduction operators which can be applied. Our aim of the partially reduced product is that it allows the user to choose the level of interaction between the component domains and therefore choose how efficient the analysis will be.

We are interested in a grid-polyhedron domain as this will be able to capture both the limit and distribution information which can take a relational form. A grid-polyhedron can be seen in Figure 1.3. The grid is illustrated by the square points and the polyhedron by the shaded area. Therefore the grid-polyhedron is the set of grid points that lie within the bounded shaded area. The simple piece of code that the grid-polyhedron was generated from is given in Example 6.3 on

Page 85.

For the grid-polyhedron domain we will show that we can specify six reduction operators to use in the partially reduced product. These operators will then give us the direct, reduced, smash, constraint, weakly tight and tight products. As both the grid and polyhedron domains have representations which encode equality information we will introduce the *constraint* product which will share the equalities between the component domains. The *smash* product can be used on any pair of abstract domains which have a "bottom" element as it shares this between the components, therefore for the case of grid-polyhedra we will pass from one domain to the other the emptyset if either are represented by this. Finally we will introduce the *weakly tight* product and *tight* product which can be thought of as "middle ground" reductions. The weakly tight product ensures that each constraint of the polyhedron representation intersects some grid point and the tight product ensures that each constraint of the polyhedron representation intersects some grid-polyhedron point. We will use the weakly tight and tight product reductions as alternatives to the traditional integer programming techniques of branch and bound and using cutting planes. As both of these traditional methods rely on using the simplex method [67, 76, 81] they have exponential complexity, whereas, we will show that we can produce an algorithm that can reduce any grid-polyhedron so that it is a weakly tight product and that our reduction has a complexity which is polynomial in the number of variables and constraints in the polyhedron representation.

As noted earlier, the intervals, bounded difference shapes and the octagon domain are all sub-domains of the polyhedra but have operations with polynomial complexity similar to that of the grid domain. Therefore we will also consider the product of a grid with each of these polyhedron sub-domains. It has been stated before that these simpler product domains can be used for applications such as checking if arrays are accessed out of bounds and if pointers or variables are accessed without being initialised [18, 33, 79], checking if executables such as web-plugins contain or perform harmful operations [16] and estimating the worst case execution time of a program [19, 34]. Also they can be used for the same applications as the grid domain, such as data dependence analysis or array reference analysis as noted in [63, 64] and [37]. We will show that our algorithm for producing a weakly tight grid-polyhedron can be used on these product sub-domains and that in certain circumstances it will produce either a tight product or a reduced product. Also as the intervals, bounded difference shapes and octagons have operations to minimise the number of constraints in their representation, our weakly tight reduction will have a complexity which is polynomial in the number of variables alone.

For each of the grid-polyhedron domains we will also consider the abstract operations. We will provide a complete set of operations together with the algorithms for each and show which operations preserve the given reduction. So, for example, we will investigate if we have a weakly tight product before an operation is performed, whether or not after the operation is performed it is still a weakly tight product. We will also investigate whether each of the reductions need to be performed before an operation so that information is not lost.

1.3 Plan of the Thesis

The rest of the thesis is structured as follows:

- Chapter 2 will introduce the terminology and notation used throughout the thesis.
- Chapter 3 will establish the domain of grids and present the two different descriptions that are used to represent the grids, as well as provide the algorithms for their conversion and reduction to a minimal or canonical representation.
- Chapter 4 presents several of the operations that the domain can perform along with their algorithms and complexities. The operations for the grid domain we have included are comparison, testing the equality of two grids, intersection, join, difference, affine image and pre-image and a covering box operation which computes the smallest non-relational grid given a relational one.
- Chapter 5 introduces the operation of widening which is required when the calculation of the fixed point fails to terminate due to the lattice not satisfying the ascending chain condition which can occur for rational grids. We also detail the weakly relational sub-domains of the grids and illustrate some of the applications of the grid domain.
- Chapter 6 introduces the partially reduced product which allows different amounts of interaction between the component domains, thus enabling different amounts of reduction. We use this product to establish the partially reduced grid-polyhedron domain, a domain which combines the grids with polyhedra. For this domain we define six different reduction choices, namely the direct, reduced, constraint, smash, weakly tight and tight product. For the weakly tight product we provide an algorithm that will move in the constraints of the polyhedron representation so that they all intersect grid points. Also we will discuss and specify several abstract operations the domain will require.
- Chapter 7 considers the partially reduced product of sub-domains of the grid-polyhedron domain. Specifically the grid-box domain (which includes the grid-interval domain), the grid-bds domain and the grid-octagon domain. For the grid-bds and grid-octagon domains we also introduce a sub-domain for each, called the bounded difference grid shape domain and the ogrid-octagon domain respectively, which require that the grid component has a weakly relational form. We also suggests several applications that the domains can be applied to.
- Finally Chapter 8 discusses the conclusions made and the hopes for the future of this work.

Chapter 2

Preliminaries

In this chapter we will introduce some of the definitions and notations from set theory, linear algebra and graph theory assumed throughout the thesis. We will also give an overview of the main concepts in abstract interpretation and the established domains, such as the polyhedral domain, that will be used in Chapter 6 as part of the product described there. Some of the definitions are based on those in mathematics textbooks [2, 69, 78].

2.1 Notation and Basic Concepts

The set of natural numbers is denoted by $\mathbb{N}$, integers by $\mathbb{Z}$, rationals by $\mathbb{Q}$ and reals by $\mathbb{R}$. The complexities we give for the different algorithms assume a unit cost for every arithmetic operation; we take the computation of the greatest common divisor of a pair of numbers $a, b \in \mathbb{Z}$ to be a single operation. Given sets X, Y and any relation $R \subseteq X \times Y$, the *image* for R on a subset A of X is $\{\, y \in Y \mid \exists x \in A \,.\, (x, y) \in R \,\}$, and the *pre-image* for R on a subset B of Y is $\{\, x \in X \mid \exists y \in B \,.\, (x, y) \in R \,\}$.

If $v, v' \in \mathbb{Z}$, then $\gcd(v, v')$ and $\operatorname{lcm}(v, v')$ denote the *greatest common divisor* and *least common multiplier*, respectively, of v, v'. We will assume that $\gcd(0, 0) = 0$. Suppose now $v, v' \in \mathbb{Q}$, so that, $v = \frac{a}{b}$ and $v' = \frac{a'}{b'}$ for some $a, b, a', b' \in \mathbb{Z}$. Then we also write

$$\gcd(v, v') := \frac{r}{s}, \qquad \text{where } s = \operatorname{lcm}(b, b') \text{ and } r = \gcd\Big(\frac{as}{b}, \frac{a's}{b'}\Big).$$

Note that the gcd is well defined as it does not depend on the choices of a, b, a', b'. Let $t, t' \in \mathbb{Z}$

be relatively prime such that $tv + t'v' = \gcd(v, v')$. Then we write

$$\mathrm{gcdext}(v, v') := \big(\gcd(v, v'), (t, t')\big).$$

Let $t, f \in \mathbb{R}$ where $f > 0$. Then

$$t \bmod f = t', \quad \text{where} \quad 0 \leq t' < f \quad \text{if} \quad \exists \mu \in \mathbb{Z},\ t = t' + \mu f.$$

Let $t, f \in \mathbb{R}$ where $f = 0$. Then $t \bmod f = t$.

2.1.1 Sets

The *cardinality* of a set S is denoted by $\# S$. If S is a set, we denote the set of non-negative elements in S by S_+. We use the shorthand notation $S[s'/s]$ for the set $\big(S \setminus \{s\}\big) \cup \{s'\}$. We will denote the *emptyset* by $\varnothing$ and the *powerset* of a set S by $\wp(S)$. We now describe the type of properties a relation may have on a set.

Definition 2.1 (Relation Properties.) *Let $\preceq$ be a binary relation on the set S. Then the relation $\preceq$ is said to be* reflexive *if $\forall s \in S, s \preceq s$. The relation $\preceq$ is said to be* symmetric *if $\forall s, t \in S$, such that $s \preceq t$ implies that $t \preceq s$. The relation $\preceq$ is said to be* anti-symmetric *if $\forall s, t \in S$, such that $s \preceq t$ and $s \neq t$ implies that $t \not\preceq s$. The relation $\preceq$ is said to be* transitive *if $\forall s, t, u \in S$, such that $s \preceq t$ and $t \preceq u$ implies that $s \preceq u$. Also the relation $\preceq$ is said to be a* partial order *if it is reflexive, anti-symmetric and transitive.*

A set S together with a partial order $\preceq$ is also said to be partially ordered, and written $\langle S, \preceq \rangle$. We will refer to $\langle S, \preceq \rangle$ as a poset.

Definition 2.2 (Total Order.) *A binary relation $\preceq$ on a set S is said to be a* total order *if $\forall s, t \in S$ either $s \preceq t$ or $t \preceq s$.*

Let $\mathbb{Q}_\infty := \mathbb{Q} \cup \{+\infty\}$ be totally ordered by the extension of '$\prec$' such that $d \prec +\infty$ for each $d \in \mathbb{Q}$.

Definition 2.3 (Least Upper Bound.) *Let $\langle S, \preceq \rangle$ be a partially ordered set and let $T \subseteq S, T \neq \varnothing$. Then $s \in S$ is the* least upper bound *(or lub) of T if*

1. *$t \preceq s$ for all $t \in T$.*
2. *When u is such that $t \preceq u$ for all $t \in T$, then $s \preceq u$.*

The *greatest lower bound* (or glb) is defined dually. Note that the lub is also called the supremum and the glb is also called the infimum.

Definition 2.4 (Lattice.) *A partially ordered set $\langle S, \preceq \rangle$ is a* lattice *if every finite subset of S has a lub and glb. A lattice is* complete *if every non-empty subset of S has a lub and glb in S.*

Definition 2.5 (Minkowski's sum.) *If $S, T \subseteq \mathbb{R}^n$, then $S + T$ denotes the* Minkowski's sum *defined by*

$$S + T := \{\, \mathbf{s} + \mathbf{t} \in \mathbb{R}^n \mid \mathbf{s} \in S, \mathbf{t} \in T \,\}.$$

Definition 2.6 (Ascending Chain Condition.) *A partially ordered set S is said to satisfy the* Ascending Chain Condition *(ACC) if all increasing chains, $s_1 \preceq s_2 \preceq s_3 \preceq \ldots$, eventually become constant. That is for some n, $s_n = s_{n+1}$ for all $n \geq 1$.*

2.1.2 Vectors and Matrices

For each $i \in \{1, \ldots, n\}$, v_i denotes the i-th component of the (column) vector $\mathbf{v} \in \mathbb{R}^n$. The empty vector (in $\mathbb{R}^0$) is denoted by ϵ. Any vector $\mathbf{v} \in \mathbb{R}^n$ is also a matrix in $\mathbb{R}^{n \times 1}$ so that it can be manipulated with the usual matrix operations of addition and multiplication, both by a scalar and by another matrix. On the other hand, it is often convenient to consider a matrix $H = (\mathbf{h}_1, \ldots, \mathbf{h}_m) \in \mathbb{R}^{n \times m}$ as a finite set of vectors $\{\mathbf{h}_1, \ldots, \mathbf{h}_m\} \subseteq \mathbb{R}^n$. For each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$, the ij-th component of a matrix $H \in \mathbb{R}^{n \times m}$ is denoted by H_{ij} and the i-th row by H_i.

Definition 2.7 (Transpose.) *The* transpose *of a matrix H, denoted by H^{T}, is the matrix whose ij-th component is the ji-th component of H.*

Definition 2.8 (Triangular Form.) *A matrix $H \in \mathbb{R}^{n \times m}$ has* upper triangular form *if $n = m$, for all $i = 1, \ldots, n$, $H_{ii} \neq 0$ and, for all j where $1 \leq j < i \leq n$, $H_{ij} = 0$. Similarly H has* lower triangular form *if $n = m$, for all $i = 1, \ldots, n$, $H_{ii} \neq 0$ and, for all j where $1 \leq i < j \leq n$, $H_{ij} = 0$.*

Definition 2.9 (Positive Definite.) *An $n \times n$ matrix D is* positive definite *if $\mathbf{x}^{\mathrm{T}} D \mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{x} \neq \mathbf{0}$.*

Definition 2.10 (Affinely Independent.) *Vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ are said to be* affinely independent *if, for $\boldsymbol{\lambda} \in \mathbb{R}^m$, the set of equations $\left\{ \sum_{i=1}^m \lambda_i \mathbf{v}_i = \mathbf{0},\ \sum_{i=1}^m \lambda_i = 0 \right\}$ has $\boldsymbol{\lambda} = \mathbf{0}$ as the only solution.*

Definition 2.11 (Scalar Product.) *The* scalar product *of $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$, denoted $\langle \mathbf{v}, \mathbf{w} \rangle$, is the real number $\mathbf{v}^{\mathrm{T}} \mathbf{w} = \sum_{i=1}^n v_i w_i$.*

Definition 2.12 (Special Vectors.) *A vector that has all its elements equal to zero is called a* zero vector *and denoted by $\mathbf{0}$. A vector with 1 in the i-th position and zeroes in every other position is called the i-th* unit vector *and is denoted by $\mathbf{e}_i$. A vector $\mathbf{v} \in \mathbb{R}^n$ is said to be* non-relational *if $\mathbf{v} = \lambda \cdot \mathbf{e}_i$ for some $\lambda \in \mathbb{R}$.*

It follows that a set of vectors is said to be non-relational if each vector in the set is non-relational.

Let $S = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\} \subseteq \mathbb{R}^n$ be a set of k vectors. For all scalars $\lambda_1, \ldots, \lambda_k \in \mathbb{R}$, the vector $\mathbf{v} = \sum_{j=1}^k \lambda_j \mathbf{v}_j$ is said to be a *linear* combination of the vectors in S. Such a combination is said to be

- an *affine* combination, if $\sum_{j=1}^{k} \lambda_j = 1$;
- an *integral* combination, if $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}$;
- an *integral affine* combination, if it is both integral and affine;
- a *positive (or conic)* combination, if $\forall j \in \{1, \ldots, k\} : \lambda_j \in \mathbb{R}_+$;
- a *convex* combination, if it is both positive and affine.

We denote by $\mathrm{affine.hull}(S)$ (resp., $\mathrm{int.hull}(S)$, $\mathrm{int.affine.hull}(S)$, $\mathrm{conic.hull}(S)$, $\mathrm{convex.hull}(S)$) the set of all the affine (resp., integer, integer affine, positive, convex) combinations of the vectors in S. We now give some definitions which are new and needed for the work on the grid domain described in Chapter 3.

Definition 2.13 (Pivot Element.) *For $\mathbf{v} \in \mathbb{R}^n$, $\mathrm{piv}_<(\mathbf{v})$ denotes the maximum index i such that $v_i \neq 0$; if $\mathbf{v} = \mathbf{0}$, we define $\mathrm{piv}_<(\mathbf{v}) := 0$. Similarly, $\mathrm{piv}_>(\mathbf{v})$ denotes the minimum index i such that $v_i \neq 0$; if $\mathbf{v} = \mathbf{0}$, we define $\mathrm{piv}_>(\mathbf{v}) := n + 1$.*

Definition 2.14 (Pivot Equivalent Vectors.) *We say two vectors are* pivot equivalent *if $\mathrm{piv}_<(\mathbf{v}) = \mathrm{piv}_<(\mathbf{v}') = k$ and $v_k = v'_k$, or if $\mathrm{piv}_>(\mathbf{v}) = \mathrm{piv}_>(\mathbf{v}') = k$ and $v_k = v'_k$, written $\mathbf{v} \Uparrow \mathbf{v}'$ and $\mathbf{v} \Downarrow \mathbf{v}'$, respectively.*

2.1.3 Congruences and Congruence Relations

For any $a, b \in \mathbb{R}$ where $a \neq 0$, we say a *divides* b, denoted by $a|b$, if, for some $m \in \mathbb{Z}$, $am = b$.

Definition 2.15 (Congruent.) *For any $a, b, f \in \mathbb{R}$, if $a - b$ is integrally divisible by f then a is said to be* congruent *to b, written $a \equiv_f b$. In the case that $f = 0$, the congruence denotes the equality $a = b$.*

Definition 2.16 (Linear Congruence Relation.) *Let $\mathbb{S}$ be either $\mathbb{Q}$ or $\mathbb{R}$. For each vector $\mathbf{a} \in \mathbb{S}^n$ and scalars $b, f \in \mathbb{S}$, the notation $\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b$ stands for the* linear congruence relation *in $\mathbb{S}^n$ defined by the set of vectors $\left\{ \mathbf{v} \in \mathbb{R}^n \;\middle|\; \exists \mu \in \mathbb{Z} \;.\; \langle \mathbf{a}, \mathbf{v} \rangle = b + \mu f \right\}$. Also when $f = 0$, the congruence relation denotes the equality $\langle \mathbf{a}, \mathbf{x} \rangle = b$. Given the congruence relation $\beta = \big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b\big)$ we say that f is the* frequency *and b is the* base value *and if $b \neq 0$, we say b is the* inhomogeneous term. *When the frequency of the congruence relation is non-zero it said to be a* proper *congruence relation.*

Provided $\mathbf{a} \neq \mathbf{0}$, the congruence relation $\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b$ defines the set of affine hyperplanes $\left\{ \big(\langle \mathbf{a}, \mathbf{x} \rangle = b + \mu f\big) \;\middle|\; \mu \in \mathbb{Z} \right\}$. The congruence $\langle \mathbf{0}, \mathbf{x} \rangle \equiv_f b$ defines the universe $\mathbb{R}^n$ if $b \equiv_f 0$, and the emptyset, otherwise. We will assume that in such a congruence (when $\mathbf{a} = \mathbf{0}$) we have $b \neq 0$. Any vector that satisfies one of the equalities $\langle \mathbf{a}, \mathbf{x} \rangle = b + \mu f$ for any $\mu \in \mathbb{Z}$ is said to *satisfy* the congruence relation $\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b$. We do not distinguish between syntactically

different congruences defining the same set of vectors in $\mathbb{S}^n$ so that, e.g., $x \equiv_1 2$ and $2x \equiv_2 4$ are considered to be the same congruence. We can now extend the non-relational and pivot notation to congruences, so that,

Definition 2.17 (Non-relational Congruence.) *A congruence $\beta = \big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b\big)$ in $\mathbb{S}^n$ is said to be* non-relational *if $\mathbf{a} = \lambda \cdot \mathbf{e}_i$ for some $\lambda \in \mathbb{S}$.*

It follows that a set of congruences is said to be non-relational if each congruence in the set is non-relational.

Definition 2.18 (Pivot Equivalent Congruences.) *If $\beta = \big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f a_0\big)$ then $\mathrm{piv}_<(\beta) := \mathrm{piv}_<(\mathbf{a})$. Also if $\gamma = \big(\langle \mathbf{c}, \mathbf{x} \rangle \equiv_g c_0\big)$ and $g\mathbf{a} \Uparrow f\mathbf{c}$, then we write $\beta \Uparrow \gamma$ and say that β and γ are* pivot equivalent congruences. *Observe that this means that β and γ are either both equalities or both proper congruences.*

2.1.4 Graph Theory

We now introduce some of the notation and terminology that will be used to describe the graphs that can encode the information of the weakly relational domains we will introduce later. This information is based on [12].

Let $\mathcal{N}$ be a finite set of *nodes*, then we will define what it is to be a rational-weighted directed graph.

Definition 2.19 (Rational-weighted Directed Graph.) *A* rational-weighted directed graph *(we say graph, for short) W in $\mathcal{N}$ is a pair $(\mathcal{N}, w)$, where $w \colon \mathcal{N} \times \mathcal{N} \to \mathbb{Q}_\infty$ is the weight function for W. Let $W = (\mathcal{N}, w)$ be a graph. A pair $(n_i, n_j) \in \mathcal{N} \times \mathcal{N}$ is an* arc *of W if $w(n_i, n_j) < +\infty$; the arc is* proper *if $n_i \neq n_j$. A* path *$\pi = n_0 \cdots n_p$ in W is a non-empty and finite sequence of nodes such that (n_{i-1}, n_i) is an arc of W, for all $i = 1, \ldots, p$. The path π is* simple *if each node occurs at most once in π. The path π is* proper *if all the arcs in it are proper.*

Let $\pi = n_0 \cdots n_p$ then, each node n_i, where $i = 0, \ldots, p$, and each arc (n_{i-1}, n_i), where $i = 1, \ldots, p$, is said to be *in* the path π. The *length* of the path π is the number p of occurrences of arcs in π and denoted by $\|\pi\|$; the *weight* of the path π is $\sum_{i=1}^{p} w(n_{i-1}, n_i)$ and denoted by $w(\pi)$. The path π is a *proper cycle* if it is a proper path, $n_0 = n_p$ and $p \geq 2$. If $\pi_1 = n_0 \cdots n_h$ and $\pi_2 = n_h \cdots n_p$ are paths, where $0 \leq h \leq p$, then the path concatenation $\pi = n_0 \cdots n_h \cdots n_p$ of π_1 and π_2 is denoted by $\pi_1 :: \pi_2$; if $\pi_1 = n_0 n_1$ (so that $h = 1$), then $\pi_1 :: \pi_2$ will also be denoted by $n_0 \cdot \pi_2$. Note that path concatenation is not the same as sequence concatenation. A graph $(\mathcal{N}, w)$ can be interpreted as the system of *potential constraints*

$$\mathcal{C} := \big\{\, n_i - n_j \leq w(n_i, n_j) \;\big|\; n_i, n_j \in \mathcal{N} \,\big\},$$

where the nodes are interpreted as the variables of the constraint and the weight function $w(n_i, n_j)$ is the constant.

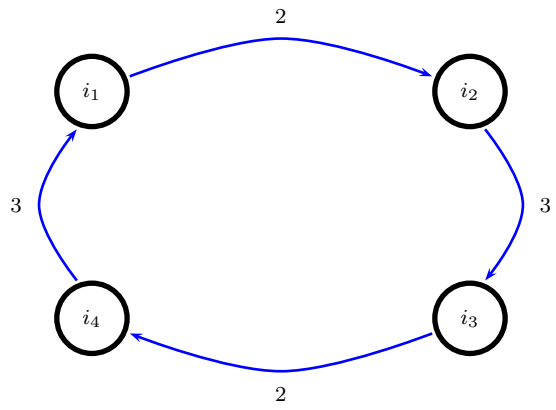


Figure 2.1: A simple graph.

Example 2.20 *Consider the graph* $(\mathcal{N}, w)$ *where* $\mathcal{N} = \{i_1, i_2, i_3, i_4\}$. *The weights for the arcs of the graph are as follows* $w(i_1, i_2) = 2, w(i_2, i_3) = 3, w(i_3, i_4) = 2, w(i_4, i_1) = 3$. *This gives the set of constraints*

$$\mathcal{C} := \big\{\, i_1 - i_2 \leq 2, i_2 - i_3 \leq 3, i_3 - i_4 \leq 2, i_4 - i_1 \leq 3 \,\big\}.$$

The graph $(\mathcal{N}, w)$ *is illustrated in Figure 2.1.*

Definition 2.21 (Consistent Graph.) *The graph* $(\mathcal{N}, w)$ *is* consistent *if and only if the system of constraints it represents is satisfiable in* $\mathbb{Q}$, *i.e., there exists a rational valuation* $\rho\colon \mathcal{N} \to \mathbb{Q}$ *such that, for each constraint* $(n_i - n_j \leq d) \in \mathcal{C}$, *the relation* $\rho(n_i) - \rho(n_j) \leq d$ *holds.*

It is well-known that a graph is consistent if and only if it has no negative weight cycle (see [24, Section 25.5]). Note that, the set of consistent graphs in $\mathcal{N}$ is denoted by $\mathbb{W}$, since the graphs will encode information about elements of a weakly relational domain. This set is partially ordered by the relation '$\trianglelefteq$' defined, for all $W_1 = (\mathcal{N}, w_1)$ and $W_2 = (\mathcal{N}, w_2)$, by

$$W_1 \trianglelefteq W_2 \quad \Longleftrightarrow \quad \forall i, j \in \mathcal{N} : w_1(i, j) \leq w_2(i, j).$$

We write $W \triangleleft W'$ when $W \trianglelefteq W'$ and $W \neq W'$. When augmented with a bottom element $\bot$ representing inconsistency, this partially ordered set becomes a non-complete lattice $\mathbb{W}_\bot = \big\langle \mathbb{W} \cup \{\bot\}, \trianglelefteq, \sqcap, \sqcup \big\rangle$, where '$\sqcap$' and '$\sqcup$' denote the finitary greatest lower bound and least upper bound operators, respectively.

Definition 2.22 (Closed graph.) *A consistent graph* $W = (\mathcal{N}, w)$ *is* closed *if the following*

properties hold:

$$\forall i \in \mathcal{N} : w(i,i) = 0; \tag{2.1}$$

$$\forall i,j,k \in \mathcal{N} : w(i,j) \leq w(i,k) + w(k,j). \tag{2.2}$$

The (shortest-path) closure *of a consistent graph W in $\mathcal{N}$ is*

$$\mathrm{closure}(W) := \bigsqcup \{\, W' \in \mathbb{W} \mid W' \trianglelefteq W \text{ and } W' \text{ is closed} \,\}.$$

Although the lattice of rational graphs is not complete, it will include the infinite least upper bound defining the closure of a rational graph W. Informally, this must hold since the weights of the least upper bound graph must be linear combinations of the rational weights of W and hence are also rational.

When trivially extended so as to behave as the identity function on the bottom element $\bot$, shortest-path closure is a kernel operator (monotonic, idempotent and reductive) on the lattice $\mathbb{W}_\bot$, therefore providing a canonical form.

The following lemma recalls a well-known result for closed graphs (for a proof, see Lemma 5 in [5]).

Lemma 2.23 *Let $W = (\mathcal{N}, w) \in \mathbb{W}$ be a closed graph. Then, for any path $\pi = i \cdots j$ in W, it holds that $w(i,j) \leq w(\pi)$.*

2.2 Abstract Interpretation

As stated in Chapter 1, abstract interpretation was introduced in 1977 by Cousot and Cousot [27]. It takes a set of possible properties of a program and approximates them by a set of intuitively descriptive abstract properties.

Definition 2.24 (Galois Connection.) *Let $\langle C, \preceq_C \rangle, \langle A, \preceq_A \rangle$ be two posets. Also let $\alpha : C \to A$ and $\gamma : A \to C$. Then a* Galois Connection *is a pair of mappings α, γ such that $\forall a_1, a_2 \in A, \forall c_1, c_2 \in C$*

$$\begin{aligned} c_1 \preceq_C c_2 &\implies \alpha(c_1) \preceq_A \alpha(c_2) \\ a_1 \preceq_A a_2 &\implies \gamma(a_1) \preceq_C \gamma(a_2) \\ \alpha(c_1) \preceq_A a_1 &\iff c_1 \preceq_C \gamma(a_1). \end{aligned}$$

The functions $\alpha : C \to A$ and $\gamma : A \to C$ are called the *abstraction* and *concretisation* functions respectively and the sets A and C are called the *abstract domain* and *concrete domain* respectively.

Definition 2.25 (Galois Insertion.) *Let $\langle C, \preceq_C \rangle, \langle A, \preceq_A \rangle$ be two posets. Also let $\alpha : C \to A$ and $\gamma : A \to C$. Then a* Galois Insertion *is a pair of mappings α, γ such that the pair are a*

Abstract Domain	Constraint Form
Interval	$b_1 \leq x_i \leq b_2$
Bounded Difference Shape	$a_i x_i - a_j x_j \leq b$ where $a_i, a_j \in \{-1, 0, 1\}$ and $a_i \neq a_j$
Two Variables Per Inequality	$a_i x_i - a_j x_j \leq b$
Octagon	$a_i x_i - a_j x_j \leq b$ where $a_i, a_j \in \{-1, 0, 1\}$
Octahedron	$a_1 x_1 + \ldots + a_n x_n \leq b$ where $a_i, a_j \in \{-1, 0, 1\}$
Polyhedron	$a_1 x_1 + \ldots + a_n x_n \leq b$

Table 2.1: Constraint representations of some abstract domains.

Galois connection and $\forall a_1, a_2 \in A$

$$a_1 \preceq_A a_2 \iff \gamma(a_1) \preceq_C \gamma(a_2).$$

Definition 2.26 (Soundness Relation.) *The concrete and abstract domains are joined by the* soundness relation σ *such that* $\sigma \in C \times A$.

This means that for the pair $(c, a) \in C \times A$ the soundness relation links the valid concrete property c with a corresponding abstract property a which has been concluded by the abstraction.

The lattice for the abstract domain of signs and its concrete counterpart can be seen in Figure 1.1 on Page 3.

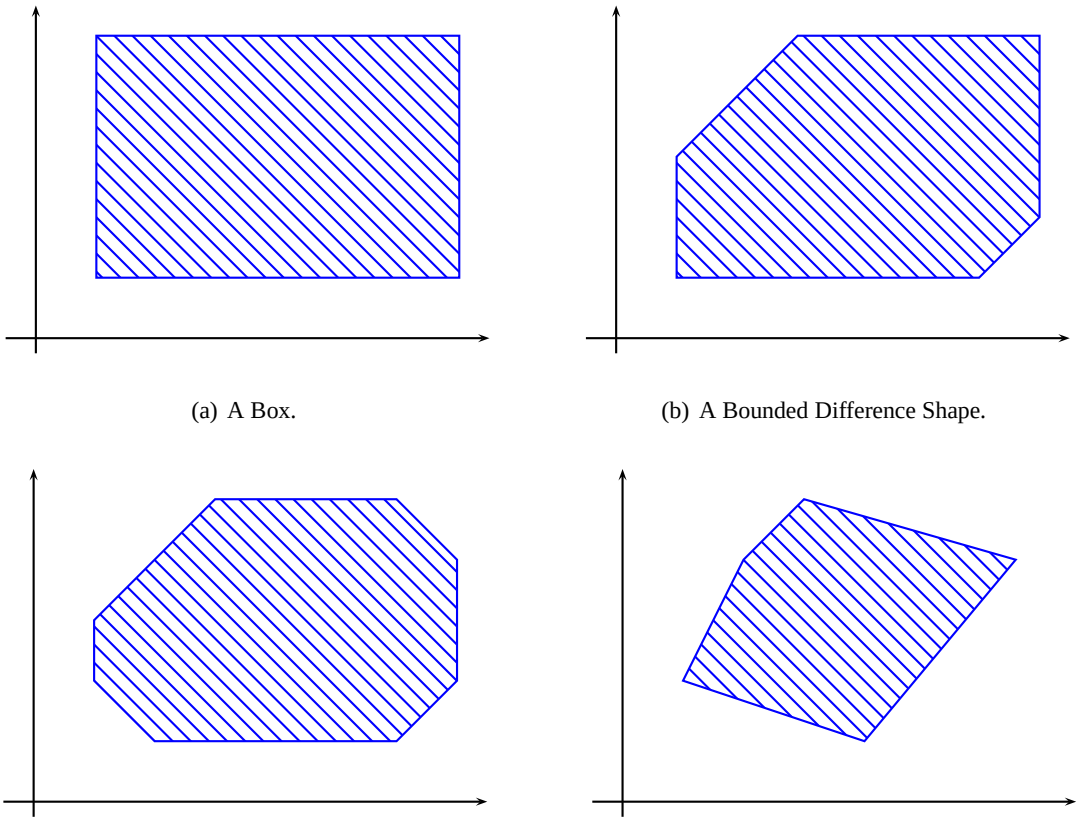
2.3 Some Numerical Domains

In the following section we will introduce some of the established abstract domains that will be considered in Chapter 6 for the grid product domains. Each of the domains considered will be a sub-domain of the polyhedron domain. There are several different polyhedron sub-domains such as the domain of convex polyhedra [6,7,9,14], octagons [4,15,50,54], octahedrons [21], bounded difference shapes (bds) [49, 51, 53], two variables per inequality (tvpi) [77] and intervals [26]. Table 2.1 shows how each of these abstract domains can be represented by different classes of constraint.

In [51], Miné introduces the term *weakly relational* when discussing the bounded difference shape domain. For the context of this thesis when we are discussing polyhedron domains we will assume that the term *weakly relational domain* refers to the bounded difference shape domain, octagon domain and the n-dimensional interval domain.

2.3.1 The Polyhedron Domain

We will now introduce some of the main features of the polyhedron domain; an illustration of a polyhedron can be seen in Figure 2.2(d). The information of this section is taken from the definitions and results of [7, 11, 14].



(a) A Box.

(b) A Bounded Difference Shape.

(c) An Octagonal Shape.

(d) A Polyhedron.

Figure 2.2: Types of Polyhedron Domain.

Definition 2.27 (Convex Polyhedra.) *The set $\mathcal{P} \subseteq \mathbb{R}^n$ is a* not necessarily closed convex polyhedron *(*NNC polyhedron*, for short) if and only if either $\mathcal{P}$ can be expressed as the intersection of a finite number of (open or closed) affine half-spaces of $\mathbb{R}^n$ or $n = 0$ and $\mathcal{P} = \varnothing$. The set of all NNC polyhedra on the vector space $\mathbb{R}^n$ is denoted $\mathbb{P}_n$. The set $\mathcal{P} \in \mathbb{P}_n$ is a* closed convex polyhedron *(*closed polyhedron*, for short) if and only if either $\mathcal{P}$ can be expressed as the intersection of a finite number of closed affine half-spaces of $\mathbb{R}^n$ or $n = 0$ and $\mathcal{P} = \varnothing$. The set of all closed polyhedra on the vector space $\mathbb{R}^n$ is denoted $\mathbb{CP}_n$. In theoretical terms, $\mathbb{P}_n$ is a* lattice *under set inclusion and $\mathbb{CP}_n$ is a* sub-lattice *of $\mathbb{P}_n$. A polyhedron, $\mathcal{P}$, is a* polytope *if $\mathcal{P}$ is bounded.*

NNC polyhedra can be specified by using two possible representations, the constraints (or implicit) representation and the generators (or parametric) representation. For the scope of this work we will only consider closed polyhedra, for a more detailed look at NNC polyhedra see [9, 14].

Constraint Representation.

Each polyhedron $\mathcal{P} \in \mathbb{CP}_n$ can be represented by a finite set of linear equality and inequality constraints $\mathcal{C}$ called a *constraint system*. We write $\mathcal{P} = \text{con}(\mathcal{C})$. By using matrix notation, we

have

$$\mathcal{P} := \{\, \mathbf{x} \in \mathbb{R}^n \mid A_1\mathbf{x} = \mathbf{b}_1, A_2\mathbf{x} \geq \mathbf{b}_2 \,\},$$

where, for all $i \in \{1, 2\}$, $A_i \in \mathbb{R}^{m_i} \times \mathbb{R}^n$ and $\mathbf{b}_i \in \mathbb{R}^{m_i}$, and $m_1, m_2 \in \mathbb{N}$ are the number of equalities and the number of non-strict inequalities, respectively. The subsets of equality and inequality constraints in system $\mathcal{C}$ are denoted by $\mathrm{eq}(\mathcal{C})$ and $\mathrm{ineq}(\mathcal{C})$, respectively.

Generator Representation.

Let $\mathcal{P} \in \mathbb{CP}_n$ be a polyhedron. Then

- a vector $\mathbf{p} \in \mathcal{P}$ is called a *point* of $\mathcal{P}$;
- a vector $\mathbf{r} \in \mathbb{R}^n$, where $\mathbf{r} \neq \mathbf{0}$, is called a *ray* (or direction of infinity) of $\mathcal{P}$ if $\mathcal{P} \neq \varnothing$ and $\mathbf{p} + \lambda\mathbf{r} \in \mathcal{P}$, for all points $\mathbf{p} \in \mathcal{P}$ and all $\lambda \in \mathbb{R}_+$;
- a vector $\mathbf{l} \in \mathbb{R}^n$ is called a *line* of $\mathcal{P}$ if both $\mathbf{l}$ and $-\mathbf{l}$ are rays of $\mathcal{P}$.

A point of a polyhedron $\mathcal{P} \in \mathbb{CP}_n$ is a *vertex* if and only if it cannot be expressed as a convex combination of any other pair of distinct points in $\mathcal{P}$. A ray $\mathbf{r}$ of a polyhedron $\mathcal{P}$ is an *extreme ray* if and only if it cannot be expressed as a positive combination of any other pair $\mathbf{r}_1$ and $\mathbf{r}_2$ of rays of $\mathcal{P}$, where $\mathbf{r} \neq \lambda\mathbf{r}_1$, $\mathbf{r} \neq \lambda\mathbf{r}_2$ and $\mathbf{r}_1 \neq \lambda\mathbf{r}_2$ for all $\lambda \in \mathbb{R}_+$ (i.e., rays differing by a positive scalar factor are considered to be the same ray).

When $\mathcal{P} \in \mathbb{CP}_n$ is a closed polyhedron, then it can be represented by finite sets of lines L, rays R and points P of $\mathcal{P}$. In this case, the 3-tuple $\mathcal{G} = (L, R, P)$ is said to be a *generator system* for $\mathcal{P}$ since we have

$$\mathcal{P} = \mathrm{linear.hull}(L) + \mathrm{conic.hull}(R) + \mathrm{convex.hull}(P),$$

where the symbol '$+$' denotes the Minkowski's sum.

For any $\mathcal{P} \in \mathbb{CP}_n$ and generator system $\mathcal{G} = (L, R, P)$ for $\mathcal{P}$, we have $\mathcal{P} = \varnothing$ if and only if $P = \varnothing$. Note that, any set of generating points P must contain all the vertices of $\mathcal{P}$. Also $\mathcal{P}$ can be non-empty and have no vertices, in this case, as P is necessarily non-empty, it must contain points of $\mathcal{P}$ that are *not* vertices. For instance, the half-space of $\mathbb{R}^2$ corresponding to the single constraint $y \geq 0$ can be represented by the generator system $\mathcal{G} = (L, R, P)$ such that $L = \{(1, 0)^{\mathrm{T}}\}$, $R = \{(0, 1)^{\mathrm{T}}\}$, and $P = \{(0, 0)^{\mathrm{T}}\}$. It is also worth noting that the only ray in R is *not* an extreme ray of $\mathcal{P}$.

When $\mathcal{P} = \mathrm{con}(\mathcal{C}) \neq \varnothing$, we say that the constraint system $\mathcal{C}$ is in *minimal form* if $\#\,\mathrm{eq}(\mathcal{C}) = n - \dim(\mathcal{P})$ and there does not exist $\mathcal{C}' \subset \mathcal{C}$ such that $\mathrm{con}(\mathcal{C}') = \mathcal{P}$. All the constraint systems in minimal form describing a given polyhedron have the same cardinality. When the constraint system $\mathcal{C}$ is not in minimal form, a constraint $\gamma \in \mathcal{C}$ is said to be *redundant* in $\mathcal{C}$ if $\mathrm{con}\big(\mathcal{C} \setminus \{\gamma\}\big) = \mathrm{con}(\mathcal{C})$.

Similarly, a generator system $\mathcal{G} = (L, R, P)$ for a polyhedron $\mathcal{P} \in \mathbb{CP}_n$ is said to be in minimal form if there does not exist a generator system $\mathcal{G}' = (L', R', P') \neq \mathcal{G}$ for $\mathcal{P}$ such that $L' \subseteq L$, $R' \subseteq R$ and $P' \subseteq P$.

Any polyhedron $\mathcal{P} \in \mathbb{CP}_n$ can be described by using a constraint system $\mathcal{C}$, a generator system $\mathcal{G}$, or both by means of the *double description pair (DD pair)* $(\mathcal{C}, \mathcal{G})$. The *double description method* [57] is a collection of novel theoretical results showing that, given one kind of representation, there are algorithms for computing a representation of the other kind and for minimising both representations by removing redundant constraints/generators.

A polyhedron is called *rational* if it can be represented by a constraint system where all the constraints have rational coefficients. It has been shown (via the double description method [57]) that a polyhedron is rational if and only if it can be represented by a generator system where all the generators have rational coefficients.

2.3.2 The Interval Domain

Interval arithmetic was introduced by Moore in [56]. It was then later introduced as a domain for use in abstract interpretation by Cousot and Cousot [26].

Definition 2.28 (Interval.) *Let $\mathbb{S} \in \{\mathbb{Q}, \mathbb{Z}\}$. A closed interval is the set of values in $\mathbb{S}$ such that $[a, b] = \{x \in \mathbb{S} | a \leq x \leq b\}$, where $\{a, b\}$ is a pair of* bounds*. We say that a is the* lower *bound and b is the* upper *bound of the interval $[a, b]$. If both the bounds are in $\mathbb{S}$, the interval is said to be* bounded*. An* integral interval *is a pair $[a, b] \in [\mathbb{Z} \cup \{\infty\}]^2$ where $a < \infty$ for all $a \in \mathbb{Z}$. Also a* rational interval *is a pair $[a, b] \in [\mathbb{Q}_\infty]^2$ where $a < \infty$ for all $a \in \mathbb{Q}$.*

Let $\mathbb{S} \in \{\mathbb{Q}, \mathbb{Z}\}$. Then the abstract domain of *intervals* in $\mathbb{S}$ is given by:

$$\mathbb{I}_\mathbb{S} := \{\bot, \top\} \cup \big\{[a, b] \;\big|\; a \in \mathbb{S} \cup \{-\infty\}, b \in \mathbb{S} \cup \{\infty\}, a \leq b\big\} \setminus \{[-\infty, \infty]\}$$

where, for all $[a_1, b_1], [a_2, b_2] \in \mathbb{I}_\mathbb{S}$,

$$[a_1, b_1] \sqsubseteq [a_2, b_2] \iff a_1 \geq a_2 \wedge b_1 \leq b_2;$$

$$[a_1, b_1] \sqcap [a_2, b_2] := \begin{cases} [a, b], & \text{if } a = \max(a_1, a_2),\ b = \min(b_1, b_2),\ a \leq b, \\ \bot, & \text{otherwise;} \end{cases}$$

$$[a_1, b_1] \sqcup [a_2, b_2] := [a, b], \text{ where } a = \min(a_1, a_2),\ b = \max(b_1, b_2),\ a \leq b.$$

We can use the interval domain to create a domain of *Boxes* over a set $\mathbb{S}$ in n dimensions, where $\mathbb{S} \in \{\mathbb{Q}, \mathbb{Z}\}$. A non-empty n-dimensional *box* $\mathcal{B}$ is a sequence $(I_1, \ldots . I_n)$ of intervals over the set $\mathbb{S}$. A subset and non-relational form of the polyhedron domain is that of intervals and boxes. An illustration of a 2-dimensional box can be seen in Figure 2.2(a).

2.3.3 The Bounded Difference Shape Domain

We now introduce the domain of bounded difference shapes and show how we will encode the constraints of the domain as weighted graphs.

Definition 2.29 (Bounded Difference Constraints.) *Let* $\mathbf{a} \in \mathbb{R}^n$ *and* $d \in \mathbb{R}$*, then for each symbol* $\bowtie \in \{=, \leq\}$*, the linear constraint* $\langle \mathbf{a}, \mathbf{v} \rangle \bowtie d$ *is said to be a* bounded difference constraint *if and only if there exists two indices* $i, j \in \{1, \ldots, n\}$ *such that*

- $a_i, a_j \in \{-1, 0, 1\}$ *and* $a_i \neq a_j$
- $a_k = 0$*, for all* $k \notin \{i, j\}$*.*

Definition 2.30 (Bounded Difference Shape.) *A convex polyhedron* $\mathcal{P} \in \mathbb{CP}_n$ *is said to be a* bounded difference shape (BDS) *if and only if* $\mathcal{P}$ *can either be expressed as the intersection of a finite number of bounded difference constraints or* $n = 0$ *and* $\mathcal{P} = \varnothing$*.*

An illustration of a 2-dimensional bounded difference shape can be seen in Figure 2.2(b). The bounded difference shapes form a weakly relational domain which extends the non-relational interval domain but is still a subset of the polyhedron domain. A finite system $\mathcal{C}$ of bounded differences on variables $\mathcal{V} = \{v_0, \ldots, v_{n-1}\}$ can be represented by a weighted directed graph $W = (\mathcal{N}_{\mathbf{0}}, w)$ where $\mathbf{0} \notin \mathcal{V}$ is the *special variable*, $\mathcal{N}_{\mathbf{0}} = \{\mathbf{0}\} \cup \mathcal{V}$, and the weight function w is defined, for each $v_i, v_j \in \mathcal{N}_{\mathbf{0}}$, by

$$w(v_i, v_j) := \begin{cases} \min\{\, d \in \mathbb{Q} \mid (v_i - v_j \leq d) \in \mathcal{C} \,\}, & \text{if } v_i \neq \mathbf{0} \text{ and } v_j \neq \mathbf{0}; \\ \min\{\, d \in \mathbb{Q} \mid (v_i \leq d) \in \mathcal{C} \,\}, & \text{if } v_i \neq \mathbf{0} \text{ and } v_j = \mathbf{0}; \\ \min\{\, d \in \mathbb{Q} \mid (-v_j \leq d) \in \mathcal{C} \,\}, & \text{if } v_i = \mathbf{0} \text{ and } v_j \neq \mathbf{0}; \\ 0, & \text{if } v_i = v_j = \mathbf{0}. \end{cases}$$

Notice that we assume that $\min \varnothing = +\infty$; moreover, unary constraints are encoded by means of the special variable, which is meant to always have value 0. We will use the definitions and notation introduced earlier for weighted directed graphs. In particular, a graph encoding a consistent system of bounded differences will be called a *bounded difference graph*.

Let $\mathcal{P} = \mathrm{con}(\mathcal{C})$ be a bounded difference shape. As we can represent a bds by a weighted graph we can apply the closure algorithm to the weighted graph to produce a closed weighted graph that represents the bds. From this we can then re-calculate the set of bounded difference constraints and represent the bds by these. So from now on we will assume that a closed set of constraints for a bds refers to the set derived from a closed weighted graph and denote this set of bounded difference constraints by $\mathrm{closure}(\mathcal{C})$.

2.3.4 The Octagon Domain

We now introduce the octagon domain and show how, like the bds domain, we can encode the constraints of the domain as weighted graphs. The following information is based on results

from [12]. For the following definition of octagonal constraints let us assume that there is a fixed set $\mathcal{V} = \{v_0, \ldots, v_{n-1}\}$ of n variables.

Definition 2.31 (Octagonal Constraints.) *Let* $\mathbf{a} \in \mathbb{R}^n$ *and* $d \in \mathbb{R}$*, then for each symbol* $\bowtie \in \{=, \leq\}$*, the linear constraint* $\langle \mathbf{a}, \mathbf{v} \rangle \bowtie d$ *is said to be an* octagonal constraint *if and only if there exists two distinct indices* $i, j \in \{1, \ldots, n\}$ *such that* $i < j$ *and*

- $a_i, a_j \in \{-1, 0, 1\}$ *and* $a_i \neq 0$
- $a_k = 0$*, for all* $k \notin \{i, j\}$*.*

Definition 2.32 (Octagon.) *A convex Polyhedron* $\mathcal{P} \in \mathbb{CP}_n$ *is said to be an* octagon *if and only if* $\mathcal{P}$ *can either be expressed as the intersection of a finite number of octagonal constraints or* $n = 0$ *and* $\mathcal{P} = \varnothing$*.*

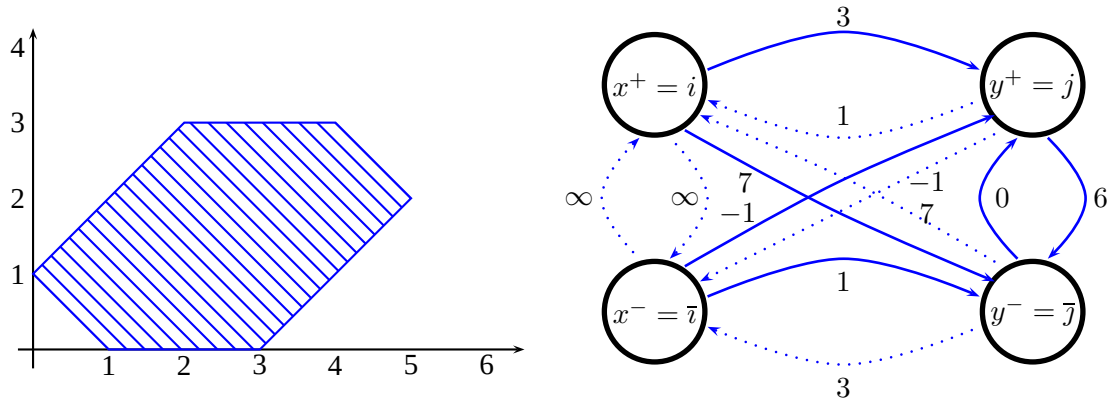
An illustration of a 2-dimensional octagon can be seen in Figure 2.2(c). The octagon domain forms a weakly relational domain which extends the weakly relational bounded difference shapes but is still a subset of the polyhedron domain. Octagonal constraints can be encoded using potential constraints by splitting each variable v_i into two separate forms: a positive form v_i^+, which we interpret as $+v_i$; and a negative form v_i^-, which we interpret as $-v_i$. Then we can write any octagonal constraint $a_i v_i + a_j v_j \leq d$ as a potential constraint $v - v' \leq d_0$ where $v, v' \in \{v_i^+, v_i^-, v_j^+, v_j^-\}$ and $d_0 \in \mathbb{Q}$. Namely, an octagonal constraint such as $v_i + v_j \leq d$ can be translated into the potential constraint $v_i^+ - v_j^- \leq d$; alternatively, the same octagonal constraint can be translated into $v_j^+ - v_i^- \leq d$. Furthermore, unary (octagonal) constraints such as $v_i \leq d$ and $-v_i \leq d$ can be encoded as $v_i^+ - v_i^- \leq 2d$ and $v_i^- - v_i^+ \leq 2d$, respectively.

From now on, we can assume that the set of nodes is $\mathcal{N}^{\pm} := \{0, \ldots, 2n-1\}$. These nodes will denote the positive and negative forms of the variables in $\mathcal{V}$: for all $i \in \mathcal{N}^{\pm}$, if $i = 2k$, then i represents the positive form v_k^+ and, if $i = 2k+1$, then i represents the negative form v_k^- of the variable v_k. To simplify the presentation, for each $i \in \mathcal{N}^{\pm}$, we let $\overline{i}$ denote $i+1$, if i is even, and $i-1$, if i is odd, so that, for all $i \in \mathcal{N}^{\pm}$, we also have $\overline{i} \in \mathcal{N}^{\pm}$ and $\overline{\overline{i}} = i$. Then we can rewrite a potential constraint $v - v' \leq d$ where $v \in \{v_k^+, v_k^-\}$ and $v' \in \{v_l^+, v_l^-\}$ as the potential constraint $i - j \leq d$ in $\mathcal{N}^{\pm}$ where, if $v = v_k^+$, $i = 2k$ and if $v = v_k^-$, $i = 2k+1$; similarly, if $v' = v_l^+$, $j = 2l$ and if $v' = v_l^-$, $j = 2l+1$.

Definition 2.33 (Octagonal graph.) *A (rational)* octagonal graph *is any consistent graph* $W = (\mathcal{N}^{\pm}, w)$ *that satisfies the coherence assumption:*

$$\forall i, j \in \mathcal{N}^{\pm} : w(i, j) = w(\overline{j}, \overline{i}). \tag{2.3}$$

The set $\mathbb{O}$ of all octagonal graphs together with the addition of the bottom element, representing an unsatisfiable system of constraints, is a sub-lattice of $\mathbb{W}_{\perp}$, sharing the same least upper bound and greatest lower bound operators. As for the bounded difference shapes, as we can represent an octagon by a weighted graph we can apply the closure algorithm to the weighted graph to produce



(a) An Octagon $\mathcal{P} = \mathrm{con}(\mathcal{C})$.

(b) A graph $(\mathcal{N}^{\pm}, w)$.

Figure 2.3: A simple octagonal graph.

a closed weighted graph that represents the octagon. From this we can then re-calculate the set of octagonal constraints and represent the octagon by these. So from now on we will assume that a closed set of constraints for an octagon refers to the set derived from a closed weighted graph.

Example 2.34 *Let $\mathcal{P} = \mathrm{con}(\mathcal{C})$ be the octagon given the set of constraints*

$$\mathcal{C} := \big\{\, 0 \le y \le 3,\ -1 \le x - y \le 3,\ 1 \le x + y \le 7 \,\big\},$$

over the set of variables $\mathcal{V}$. Then $\mathcal{P}$ can be seen in Figure 2.3(a). We can split each variable into its positive and negative form to get an alternative set of constraints

$$\mathcal{C} := \big\{\, y^+ - y^- \le 6,\ y^- - y^+ \le 0,\ x^+ - y^+ \le 3,\ x^+ - y^- \le 7,\ x^- - y^+ \le -1,\ x^- - y^- \le 1 \,\big\}.$$

Then from this set of constraints we can consider the graph $(\mathcal{N}^{\pm}, w)$ where $\mathcal{N}^{\pm} = \{i, \overline{i}, j, \overline{j}\}$. The weights for the arcs of the graph derived from the constraints are as follows

$$w(j,\overline{j}) = 6,\ w(\overline{j},j) = 0,\ w(i,j) = 3,\ w(i,\overline{j}) = 7,\ w(\overline{i},j) = -1,\ w(\overline{i},\overline{j}) = 1$$

the weights for the arcs derived from the coherence assumption are

$$w(\overline{j},\overline{i}) = 3,\ w(\overline{j},i) = -1,\ w(j,\overline{i}) = 7,\ w(j,i) = 1$$

and for all other arcs the weight is ∞. The graph $(\mathcal{N}^{\pm}, w)$ can be seen in Figure 2.3(b), where the arcs derived from the constraints are given by the solid lines and all others are given by the dashed lines.

Chapter 3

The Grid Domain

3.1 Introduction

The purpose of this chapter is to introduce the domain of rational grids and their two representations, together with an algorithm for converting between these two representations. The grid domain will interpret information from programs as sets of equally spaced points. We will demonstrate how we can infer information about the pattern of values a variable can take from program fragments, see Examples 3.2, 3.11 and 3.12. We will then show how these two representations form the two components of a double description method for the grid domain very similar to that for convex polyhedra [57].

The first representation we will introduce is that of the congruence system. This system introduces relations of the form $\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b$ which stands for the set of vectors

$$\{\, \mathbf{x} \in \mathbb{R}^n \mid \exists \mu \in \mathbb{Z} \,.\, \langle \mathbf{a}, \mathbf{x} \rangle = b + \mu f \,\}.$$

3.2 The Congruence Representation

A *congruence system in* $\mathbb{Q}^n$ is a finite set of congruence relations $\mathcal{C}$ in $\mathbb{Q}^n$. As we do not distinguish between syntactically different congruences defining the same set of vectors, we can assume that all proper congruences in $\mathcal{C}$ have modulus 1.

Definition 3.1 (Rational Grid.) *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$. If $\mathcal{L}$ is the set of vectors in $\mathbb{R}^n$ that satisfy all the congruences in $\mathcal{C}$, we say that $\mathcal{L}$ is a* grid described by *a congruence system $\mathcal{C}$ in $\mathbb{Q}^n$. We also say that $\mathcal{C}$* is a congruence system for *$\mathcal{L}$ and write $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$. If $\mathrm{gcon}(\mathcal{C}) = \varnothing$, then we say that $\mathcal{C}$ is* inconsistent.

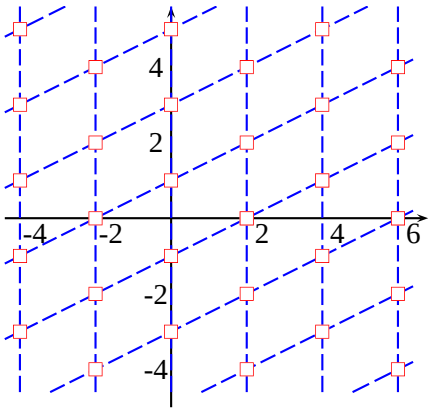


Figure 3.1: A grid in $\mathbb{R}^2$ represented by a congruence system.

We will now give some examples of grids. The first example shows how we can take part of a program and infer from it the distribution information.

Example 3.2 *Consider first the following program fragment (based on an example in [32]) for any value of* `m`*:*

```
x := 2; y := 0;
for i := 1 to m
  if ... then
    x := x + 4
  else
    x := x + 2; y := y + 1
  endif
endfor
```

if we consider the distribution of possible values of integer variables x and y resulting from the execution of the code, we obtain the following congruence relations $x \equiv_2 0$ and $-x + 2y \equiv_4 2$. The grid is illustrated in Figure 3.1 by the square points and the congruences that produce the grid are shown by the dashed lines.

The Example 3.3 shows two different ways of representing an empty grid using the congruence system.

Example 3.3 *Consider the congruence systems $\{\langle \mathbf{0}, \mathbf{x}\rangle \equiv_0 1\}$ and $\{\langle \mathbf{a}, \mathbf{x}\rangle \equiv_2 0, \langle \mathbf{a}, \mathbf{x}\rangle \equiv_2 1\}$, for any $\mathbf{a} \in \mathbb{Q}^n$, both describe the empty grid in $\mathbb{R}^n$. In fact, the first congruence system requires that $0 = 1$, while the second one requires that the value of an expression is both even and odd, so that they are both inconsistent.*

Definition 3.4 (Grid Domain.) *The* grid domain $\mathbb{G}_n$ *is the set of all grids in $\mathbb{R}^n$ ordered by the set inclusion relation, so that $\varnothing$ and $\mathbb{R}^n$ are the bottom and top elements of $\mathbb{G}_n$ respectively.*

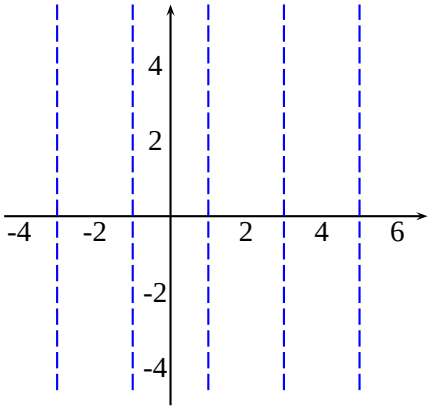


Figure 3.2: A grid in $\mathbb{R}^2$ represented by a single congruence.

Definition 3.5 (Universe Grid.) *The vector space $\mathbb{R}^n$ is called the* universe *grid.*

In set theoretical terms, $\mathbb{G}_n$ is a *lattice* under set inclusion. The *space dimension* of a grid $\mathcal{L} \in \mathbb{G}_n$ is the dimension $n \in \mathbb{N}$ of the corresponding vector space $\mathbb{R}^n$. If the maximum number of affinely independent points in $\mathcal{L}$ is $k+1$, then $\dim(\mathcal{L}) = k$ denotes the *affine dimension* of $\mathcal{L}$. The affine dimension of an empty grid is defined to be 0. Thus we have $0 \leq \dim(\mathcal{L}) \leq n$.

Example 3.6 *Consider the grid $\mathcal{L} \in \mathbb{G}_2$, which can be seen in Figure 3.2, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$ and*

$$\mathcal{C} := \{x \equiv_2 1\}.$$

Then $\dim(\mathcal{L}) = 2$ even though $\mathcal{L}$ is only represented by one congruence.

Let $\mathcal{C}$ be a congruence system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$. Suppose also that the congruence relation $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big)$ is such that $\mathcal{L}_\beta = \mathrm{gcon}\big(\{\beta\}\big)$. We say that

- $\mathcal{L}$ *is disjoint* from β if $\mathcal{L} \cap \mathcal{L}_\beta = \varnothing$; that is, adding β to $\mathcal{C}$ gives us the empty grid.
- $\mathcal{L}$ *strictly intersects* β if $\mathcal{L} \cap \mathcal{L}_\beta \neq \varnothing$ and $\mathcal{L} \cap \mathcal{L}_\beta \subset \mathcal{L}$; that is, adding β to $\mathcal{C}$ gives us a non-empty grid strictly smaller than $\mathcal{L}$.
- $\mathcal{L}$ *is included* in β if $\mathcal{L} \subseteq \mathcal{L}_\beta$; that is, adding β to $\mathcal{C}$ leaves $\mathcal{L}$ unchanged.

Example 3.7 *Consider again the grid from Example 3.2. Let*

$$\begin{aligned}\beta_1 &= (x \equiv_2 1),\\ \beta_2 &= (x \equiv_4 2),\\ \beta_3 &= (x + 2y \equiv_4 2).\end{aligned}$$

Then $\mathcal{L}$ is disjoint from β_1, $\mathcal{L}$ strictly intersects β_2 and $\mathcal{L}$ is included in β_3.

As an alternative to the congruence system we now introduce a different way to represent the grid domain, that is, by a set of generating vectors.

3.3 The Generator Representation

Let $\mathcal{L}$ be a grid in $\mathbb{G}_n$. Then

- a vector $\mathbf{p} \in \mathcal{L}$ is called a *point* of $\mathcal{L}$;
- a vector $\mathbf{q} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is called a *parameter* of $\mathcal{L}$ if $\mathcal{L} \neq \varnothing$ and $\mathbf{p} + \mu\mathbf{q} \in \mathcal{L}$, for all points $\mathbf{p} \in \mathcal{L}$ and all $\mu \in \mathbb{Z}$;
- a vector $\boldsymbol{\ell} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is called a *line* of $\mathcal{L}$ if $\mathcal{L} \neq \varnothing$ and $\mathbf{p} + \lambda\boldsymbol{\ell} \in \mathcal{L}$, for all points $\mathbf{p} \in \mathcal{L}$ and all $\lambda \in \mathbb{R}$.

If L, Q and P are finite sets of vectors in $\mathbb{Q}^n$ and

$$\mathcal{L} := \mathrm{linear.hull}(L) + \mathrm{int.hull}(Q) + \mathrm{int.affine.hull}(P),$$

then $\mathcal{L} \in \mathbb{G}_n$ is a grid (see [76, Section 4.4] and also Proposition 3.30). The 3-tuple $\mathcal{G} = (L, Q, P)$, where L, Q and P, all in $\mathbb{Q}^n$, denote sets of lines, parameters and points, respectively, is said to be a *generator system* for $\mathcal{L}$ and we write $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$; also, for convenience, we let $\mathrm{ggen}(L, Q, P)$ denote $\mathrm{ggen}(\mathcal{G})$ (without the extra parentheses). Note that the grid $\mathcal{L} = \mathrm{ggen}(L, Q, P) = \varnothing$ if and only if the set of points $P = \varnothing$. If $P \neq \varnothing$, then $\mathcal{L} = \mathrm{ggen}(L, \varnothing, Q_{\mathbf{p}} \cup P)$ where, for some $\mathbf{p} \in P$, $Q_{\mathbf{p}} = \{\, \mathbf{p} + \mathbf{q} \in \mathbb{Q}^n \mid \mathbf{q} \in Q \,\}$. As indicated in [76, Section 4.4], both congruence and generator systems can be used to describe a grid.

Proposition 3.8 *Let $\mathcal{L} \subseteq \mathbb{R}^n$. Then $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$, for some congruence system $\mathcal{C}$ in $\mathbb{R}^n$, if and only if $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$, for some generator system $\mathcal{G}$ in $\mathbb{R}^n$.*

This also follows directly from Propositions 3.29 and 3.30 in Section 3.5 which provide algorithms for converting between the two systems.

Definition 3.9 (Rectilinear Grid.) *We say that a grid $\mathcal{L}$ is* rectilinear *if it can be represented by a non-relational set of congruences or generators.*

Example 3.10 *Let $\mathcal{L} \in \mathbb{G}_2$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$, $\mathcal{C} := \{x \equiv_2 0,\ y \equiv_3 0\}$ and*

$$\mathcal{G} := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right).$$

Then $\mathcal{L}$ is a rectilinear grid.

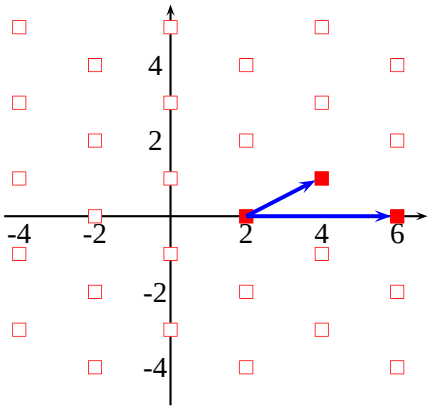


Figure 3.3: A grid in $\mathbb{R}^2$ represented by a generator system.

The following examples show how we can infer the generator descriptions for grids from fragments of programs.

Example 3.11 *Recall the simple code given in Example 3.2. The filled squares in Figure 3.3 represent the points*

$$\mathbf{p}_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \ \mathbf{p}_2 = \begin{pmatrix} 6 \\ 0 \end{pmatrix} \text{ and } \mathbf{p}_3 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

while all the squares (both filled and unfilled) in the diagram mark the position vectors $\mathbf{v} = \pi_1\mathbf{p}_1 + \pi_2\mathbf{p}_2 + \pi_3\mathbf{p}_3$*, where* $\pi_1, \pi_2, \pi_3 \in \mathbb{Z}$ *and* $\pi_1 + \pi_2 + \pi_3 = 1$*. The set of points* $P = \{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3\}$ *will* generate *the grid* $\mathcal{L} = \mathrm{ggen}(\mathcal{G}_1) = \mathrm{ggen}(\varnothing, \varnothing, P)$*. Some of these generating points can be replaced by parameters that give the direction and spacing for the neighbouring points. Specifically, by subtracting one of the points from each of the other two generating points we can obtain the parameters thus, by subtracting the point* $\mathbf{p}_1$ *from each of the points* $\mathbf{p}_2, \mathbf{p}_3$*, we obtain*

$$\mathbf{q}_2 = \begin{pmatrix} 4 \\ 0 \end{pmatrix} \text{ and } \mathbf{q}_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

which are marked by the thick lines between points $\mathbf{p}_1$ *and* $\mathbf{p}_2$ *and points* $\mathbf{p}_1$ *and* $\mathbf{p}_3$*, respectively. It follows that each point* $\mathbf{v} \in \mathcal{L}$ *can be written as* $\mathbf{v} = \mathbf{p}_1 + \pi_2\mathbf{q}_2 + \pi_3\mathbf{q}_3$ *for some* $\pi_2, \pi_3 \in \mathbb{Z}$*; the set* $Q = \{\mathbf{q}_2, \mathbf{q}_3\}$ *is called a* parameter set *for* $\mathcal{L} = \mathrm{ggen}(\mathcal{G}_2) = \mathrm{ggen}(\varnothing, Q, \{\mathbf{p}_1\})$*.*

Example 3.12 *Consider the procedure given in Figure 3.4 which is the running example of [61]. The effect of calling* q *with the pair of variables* (x, y) *set to the pair of values* (a, b) *will be to bind the vector* $(x, y)^{\mathrm{T}}$ *to the vectors* $(a, b)^{\mathrm{T}}$*,* $(15a, 18a + b)^{\mathrm{T}}$*,* $(225a, 282a + b)^{\mathrm{T}}(3375a, 4224a + b)^{\mathrm{T}}$ *and so on. Computing q(a, b) the vectors are generated as follows:*

```
q(var x, var y)
  if ...  then
    x := 3*x                    (P1)
    y := x + y                  (P2)
    q(x, y)                     (P3)
    x := 5*x                    (P4)
    y := x + y                  (P5)
  endif
```

Figure 3.4: A recursive procedure.

- *(iteration 1):*

  1. *(iteration 1.1) the if condition on the first line of the code fails so* $q(a, b) = (a, b)$.
  2. *(iteration 1.2) the if condition with parameters* (a, b) *succeeds. Then lines* `P1-P5` *must be executed:*
     (iteration 1.2) after `(P1)`*:* $(x, y) = (3a, b)$
     (iteration 1.2) after `(P2)`*:* $(x, y) = (3a, 3a + b)$
     (iteration 1.2) after `(P3)`*: the value of* $q(3a, 3a + b)$.
     So the computation continues with $q(3a, 3a + b)$

- *(iteration 2):*

  1. *(iteration 2.1) the if condition on the first line of the code fails and* $q(3a, 3a + b) = (3a, 3a + b)$.
     Now iteration 1.2 can be completed:
     (iteration 1.2) after `(P4)`*:* $(x, y) = (15a, 3a + b)$
     (iteration 1.2) after `(P5)`*:* $(x, y) = (15a, 18a + b)$.
     So $q(a, b) = (15a, 18a + b)$.
  2. *(iteration 2.2) the if condition with parameters* $(3a, 3a + b)$ *succeeds. Then lines* `P1-P5` *must be executed:*
     (iteration 2.2) after `(P1)`*:* $(x, y) = (9a, 3a + b)$
     (iteration 2.2) after `(P2)`*:* $(x, y) = (9a, 12a + b)$
     (iteration 2.2) after `(P3)`*: the value of* $q(9a, 12a + b)$.
     So the computation continues with $q(9a, 12a + b)$

- *(iteration 3):*

  1. *(iteration 3.1) the if condition on the first line of the code fails so that* $q(9a, 12a+b) = (9a, 12a + b)$.

Now iteration 2.2 can be completed:
(iteration 2.2) after `(P4)`*:* $(x, y) = (45a, 12a + b)$
(iteration 2.2) after `(P5)`*:* $(x, y) = (45a, 57a + b)$.
Now iteration 1.2 can be completed:
(iteration 1.2) after `(P4)`*:* $(x, y) = (225a, 57a + b)$
(iteration 1.2) after `(P5)`*:* $(x, y) = (225a, 282a + b)$.
So $q(a, b) = (225a, 282a + b)$.

2. *(iteration 3.2) the if condition on the first line of the code succeeds. Then lines* `P1-P5` *must be executed:*
(iteration 3.2) after `(P1)`*:* $(x, y) = (27a, 12a + b)$
(iteration 3.2) after `(P2)`*:* $(x, y) = (27a, 39a + b)$
(iteration 3.2) after `(P3)`*: the value of* $q(27a, 39a + b)$.
So the computation continues with $q(27a, 39a + b)$

- *(iteration 4):*

1. *(iteration 4.1) the if condition on the first line of the code fails so that* $q(27a, 39a + b) = (27a, 39a + b)$.
Now iteration 3.2 can be completed:
(iteration 3.2) after `(P4)`*:* $(x, y) = (135a, 39a + b)$
(iteration 3.2) after `(P5)`*:* $(x, y) = (135a, 174a + b)$.
Now iteration 2.2 can be completed:
(iteration 2.2) after `(P4)`*:* $(x, y) = (675a, 174a + b)$
(iteration 2.2) after `(P5)`*:* $(x, y) = (675a, 849a + b)$.
Now iteration 1.2 can be completed:
(iteration 1.2) after `(P4)`*:* $(x, y) = (3375a, 849a + b)$
(iteration 1.2) after `(P5)`*:* $(x, y) = (3375a, 4224a + b)$.
So $q(a, b) = (3375a, 4224a + b)$.

and so on...

Note that using integral grids, without knowing the values a *and* b*, we cannot perform any of the grid operations, described in Chapter 4, since all the values are parametric on the pair* (a, b)*. We therefore need a grid where the values capture the effect of the procedure but do not refer explicitly to the values* (a, b)*. Since the effect of the procedure in the case* $a = 0$ *is trivial, we assume* $a \neq 0$*. Consider a grid with variables* (u, v, w) *where* $u = \frac{x}{a}$*,* $v = \frac{y}{a} - \frac{b}{a}$ *and* $w = \frac{b}{a}$*. Then* $(1, 0, \frac{b}{a})^{\mathrm{T}}$ *will be the initial vector for* (u, v, w) *in any call to* `q`*. This will result in a vector of values represented as a point in a grid* $\mathcal{L}_i = \mathrm{ggen}(L, \varnothing, P_i)$*, where* i *is the number of iterations through the body of the procedure; the singleton set of lines* $L = \{(0, 0, 1)^{\mathrm{T}}\}$ *represents the fact that there is no information about the initial value* $\frac{b}{a}$ *for* w *and, for the first four iterations, the sets of points are given by* $P_0 := \{(1, 0, 0)^{\mathrm{T}}\}$*,* $P_1 := \{(15, 18, 0)^{\mathrm{T}}\}$*,* $P_2 := \{(225, 282, 0)^{\mathrm{T}}\}$

and $P_3 := \{(3375, 4224, 0)^{\mathrm{T}}\}$. *Letting* $\mathcal{L} := \mathcal{L}_0 \oplus \mathcal{L}_1 \oplus \mathcal{L}_2 \oplus \mathcal{L}_3$, *where* $\oplus$ *is the operation of grid join described in Section 4.4, we have* $\mathcal{L} = \mathrm{ggen}(L, \varnothing, P)$ *where*

$$P = \{(1,0,0)^{\mathrm{T}}, (15,18,0)^{\mathrm{T}}, (225,282,0)^{\mathrm{T}} (3375,4224,0)^{\mathrm{T}}\}.$$

Converting this to the congruence representation, using methods described in Section 3.5, we obtain

$$\mathcal{L} = \mathrm{gcon}(\{u \equiv_{14} 1, v \equiv_6 0\}).$$

This grid $\mathcal{L}$ *represents a fixpoint for the procedure; thus it includes all the possible values for the vector* $(x, y)^{\mathrm{T}}$ *that might be obtained as a result of calling* q. *If the procedure is called with* $x = a$ *where* $a \neq 0$ *and* $y = b$, *then all the possible values for the vector* $(x, y)^{\mathrm{T}}$ *are represented by the grid*

$$\mathrm{gcon}(\{x \equiv_{14a} a, y \equiv_{6a} b\}).$$

3.4 Homogeneous Form

In this section, we describe an accessible and appropriate way to represent internally the congruence and generator systems in terms of arrays (i.e., matrices) which will be required by our conversion algorithm.

Definition 3.13 (Homogeneous.) *A congruence system* $\mathcal{C}$ *is* homogeneous *if, for all* $\big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b\big) \in \mathcal{C}$, *we have* $b = 0$. *Similarly, a generator system* (L, Q, P) *is* homogeneous *if* $\mathbf{0} \in P$.

For the conversion between the two systems (described in Section 3.5) and the implementation within the PPL [13], it is convenient to work with a homogeneous system. Thus we will first convert any congruence or generator system in $\mathbb{Q}^n$ to a homogeneous system in $\mathbb{Q}^{n+1}$. The extra dimension is denoted with a 0 subscript so that the vector $\hat{\mathbf{x}}$ is given by $(x_0, \ldots, x_n)^{\mathrm{T}}$ and $\mathbf{e}_0$ denotes the vector $(1, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}$.

Consider the congruence system $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$ in $\mathbb{Q}^n$, where $\mathcal{E}$ is a set of equalities and $\mathcal{F}$ is a set of proper congruences. Then the *homogeneous form* for $\mathcal{C}$ is the congruence system $\hat{\mathcal{C}} = \hat{\mathcal{E}} \cup \hat{\mathcal{F}}$ in $\mathbb{Q}^{n+1}$ defined by:

$$\hat{\mathcal{E}} := \Big\{ \big\langle (-b, \mathbf{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\mathbf{x}} \big\rangle = 0 \;\Big|\; \big(\langle \mathbf{a}, \mathbf{x} \rangle = b\big) \in \mathcal{E} \Big\}, \tag{3.1}$$

$$\hat{\mathcal{F}} := \Big\{ \big\langle f^{-1}(-b, \mathbf{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\mathbf{x}} \big\rangle \equiv_1 0 \;\Big|\; \big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b\big) \in \mathcal{F} \Big\} \cup \big\{ \langle \mathbf{e}_0, \hat{\mathbf{x}} \rangle \equiv_1 0 \big\}. \tag{3.2}$$

The congruence $\langle \mathbf{e}_0, \hat{\mathbf{x}} \rangle \equiv_1 0$ expresses the fact that $1 \equiv_1 0$. By writing $\hat{\mathcal{E}} = (E^{\mathrm{T}}\mathbf{x} = \mathbf{0})$ and $\hat{\mathcal{F}} = (F^{\mathrm{T}}\mathbf{x} \equiv_1 \mathbf{0})$, where $E, F \subseteq \mathbb{Q}^{n+1}$, it can be seen that the pair (F, E), called the *matrix form* of $\hat{\mathcal{C}}$, is sufficient to determine $\mathcal{C}$.

Consider next a generator system $\mathcal{G} = (L, Q, P)$ in $\mathbb{Q}^n$. Then the *homogeneous form* for $\mathcal{G}$ is the generator system $\hat{\mathcal{G}} := \big(\hat{L}, \hat{Q} \cup \hat{P}, \{\mathbf{0}\}\big)$ in $\mathbb{Q}^{n+1}$ where

$$\hat{L} := \big\{(0, \boldsymbol{\ell}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \boldsymbol{\ell} \in L\big\},\ \hat{Q} := \big\{(0, \mathbf{q}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \mathbf{q} \in Q\big\},\ \hat{P} := \big\{(1, \mathbf{p}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \mathbf{p} \in P\big\}. \tag{3.3}$$

The original grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$ (resp., $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$) can be recovered from the grid $\hat{\mathcal{L}} = \mathrm{gcon}(\hat{\mathcal{C}})$ (resp., $\hat{\mathcal{L}} = \mathrm{ggen}(\hat{\mathcal{G}})$) since $\mathcal{L} = \big\{\, \mathbf{v} \in \mathbb{R}^n \;\big|\; (1, \mathbf{v}^{\mathrm{T}})^{\mathrm{T}} \in \hat{\mathcal{L}}) \,\big\}$.

Example 3.14 *Consider the grid $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$ where*

$$\mathcal{G} := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right).$$

Then the homogeneous form for $\mathcal{G}$ is $\hat{\mathcal{G}}$ where

$$\hat{\mathcal{G}} := \left(\varnothing, \begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\right).$$

3.5 Reduction and Conversion Algorithms

Many of the algorithms given for the operations on grids discussed later will require that the congruence systems not only have minimal cardinality but also that the coefficients of (a permutation of) the matrix form for the congruences can form a triangular matrix.

Definition 3.15 (Congruence System in Minimal Form.) *Suppose $\mathcal{C}$ is a congruence system in $\mathbb{Q}^n$. Then we say that $\mathcal{C}$ is in* minimal form *if either $\mathcal{C} = \{\langle \mathbf{0}, \mathbf{x}\rangle \equiv_0 1\}$ or, for each congruence $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}$, the following hold:*

1. *if $\mathrm{piv}_<(\beta) = k$, then $k > 0$ and $a_k > 0$;*
2. *for all $\beta' \in \mathcal{C} \setminus \{\beta\}$, $\mathrm{piv}_<(\beta') \neq \mathrm{piv}_<(\beta)$.*

Lemma 3.16 *If $\mathcal{C} \neq \{\langle \mathbf{0}, \mathbf{x}\rangle \equiv_0 1\}$ is a congruence system in minimal form, then $\mathcal{C}$ is consistent.*

Proof. Since $\mathcal{C}$ is in minimal form, the set of equalities $\mathcal{E} = \big\{\langle \mathbf{v}, \mathbf{x}\rangle = b \;\big|\; \langle \mathbf{v}, \mathbf{x}\rangle \equiv_f b \in \mathcal{C}\big\}$ is linearly independent; and hence $\mathcal{E}$ has a solution $\mathbf{p}$. Moreover, since, for each congruence $\big(\langle \mathbf{v}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}$ there is a corresponding equality $\big(\langle \mathbf{v}, \mathbf{x}\rangle = b\big) \in \mathcal{E}$, $\mathrm{gcon}(\mathcal{E}) \subseteq \mathrm{gcon}(\mathcal{C})$. Thus $\mathbf{p} \in \mathrm{gcon}(\mathcal{C})$ and, hence, $\mathcal{C}$ is consistent. □

We will now show how to produce an n-dimensional grid in minimal form given that it is represented by a congruence system which consists of m congruences.

Proposition 3.17 *There exists an algorithm that, for each congruence system $\mathcal{C}$ in $\mathbb{Q}^n$, computes a congruence system $\mathcal{C}'$ in minimal form such that* $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}')$. *Letting $m := \#\mathcal{C}$, the algorithm has a worst-case complexity given by* $\mathrm{O}\big(mn \min\{m, n\}\big)$.

Proof. To prove the result, we first define the key transformation step in the algorithm and show that the resulting congruence system describes the same grid. Suppose there exist distinct congruences

$$\beta_1 = \big(\langle \mathbf{a}_1, \mathbf{x}\rangle \equiv_{f_1} b_1\big), \qquad \beta_2 = \big(\langle \mathbf{a}_2, \mathbf{x}\rangle \equiv_{f_2} b_2\big) \tag{3.4}$$

in $\mathcal{C}$ such that $\operatorname{piv}_<(\beta_1) = \operatorname{piv}_<(\beta_2) = i > 0$. We will define the congruences

$$\beta_1'' = \big(\langle \mathbf{a}_1'', \mathbf{x}\rangle \equiv_{f_1} b_1''\big), \qquad \beta_2'' = \big(\langle \mathbf{a}_2'', \mathbf{x}\rangle \equiv_{f_2} b_2''\big)$$

and a congruence system $\mathcal{C}''$ such that either $\mathcal{C}'' = \big(\mathcal{C} \setminus \{\beta_1, \beta_2\}\big) \cup \{\beta_1'', \beta_2''\}$ or $\mathcal{C}'' = \big(\mathcal{C} \setminus \{\beta_1, \beta_2\}\big) \cup \{\beta_1''\}$ and show that $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}'')$. We show that $\operatorname{piv}_<(\mathbf{a}_1'') = i$ and, if β_2'' is defined, then $\operatorname{piv}_<(\mathbf{a}_2'') < i$. There are two cases.

1. At least one of β_1, β_2 is an equality; without loss of generality, we assume that β_1 is an equality so that $f_1 = 0$. Then we let $\beta_1'' = \beta_1$ and, using Gaussian elimination,

$$\mathbf{a}_2'' = \mathbf{a}_2 - (a_{2i}/a_{1i})\mathbf{a}_1, \qquad b_2'' = b_2 - (a_{2i}/a_{1i})b_1.$$

2. Both β_1 and β_2 are proper congruences; so that we can assume that $f_1 = f_2 = 1$. Let $\operatorname{gcdext}(a_{1i}, a_{2i}) = \big(r, (s, t)\big)$ and

$$\begin{aligned} \mathbf{a}_1'' &= s\mathbf{a}_1 + t\mathbf{a}_2, & b_1'' &= sb_1 + tb_2, \\ \mathbf{a}_2'' &= (-a_{2i}/r)\mathbf{a}_1 + (a_{1i}/r)\mathbf{a}_2, & b_2'' &= (-a_{2i}/r)b_1 + (a_{1i}/r)b_2. \end{aligned}$$

In both cases, let

$$\mathcal{C}'' = \begin{cases} \big(\mathcal{C} \setminus \{\beta_1, \beta_2\}\big) \cup \{\beta_1'', \beta_2''\}, & \text{if } \mathbf{a}_2'' \neq \mathbf{0} \text{ or } b_2'' \neq 0; \\ \big(\mathcal{C} \setminus \{\beta_1, \beta_2\}\big) \cup \{\beta_1''\}, & \text{otherwise.} \end{cases}$$

Then $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}'')$. Note that these transformations require a computation for each coefficient of the considered congruences so that their complexity is $\mathrm{O}\big(n\big)$.

The proof of the result is by induction on i; where $0 \leq i \leq n$ is the maximum value for which there exist distinct congruences β_1 and $\beta_2 \in \mathcal{C}$ defined as in (3.4) such that $\operatorname{piv}_<(\beta_1) = \operatorname{piv}_<(\beta_2) = i$.

The base case is when $i = 0$ so that $\mathbf{a}_1 = \mathbf{a}_2 = \mathbf{0}$. In this case, if there exists $\big(\langle \mathbf{0}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}$ and $b \equiv_f 0$ is inconsistent, let $\mathcal{C}' = \big\{\langle \mathbf{0}, \mathbf{x}\rangle \equiv_0 1\big\}$; otherwise, let $\mathcal{C}' = \big\{\, \beta \in \mathcal{C} \;\big|\; \operatorname{piv}_<(\beta) \neq$

$0\,\}$.

For the step case, we keep applying the transformation (1) if either $f_1 = 0$ or $f_2 = 0$, and (2) otherwise until no more transformations are applicable for this index; that is when we obtain a congruence system $\mathcal{C}_j$ for which $j < i$ is the maximum index such that there exist distinct congruences β_1 and $\beta_2 \in \mathcal{C}_j$ where $\mathrm{piv}_<(\beta_1) = \mathrm{piv}_<(\beta_2) = j$. We note that we will have to perform these transformations at most m times for each step, where $\#\,\mathcal{C} = m$, so that the complexity of each step is $\mathrm{O}\big(nm\big)$. By the inductive hypothesis, we can compute $\mathcal{C}'$ in minimal form such that $\mathrm{gcon}(\mathcal{C}') = \mathrm{gcon}(\mathcal{C}_j)$. Therefore $\mathrm{gcon}(\mathcal{C}') = \mathrm{gcon}(\mathcal{C})$. As we iterate at most $\min\{m, n\}$ times over the step case, it can be seen that the algorithm has complexity $\mathrm{O}\big(mn\min\{m, n\}\big)$. $\square$

Note that the algorithm mentioned in Proposition 3.17, is based on the Hermite normal form algorithm [67, 76].

As for congruence systems, for many operations and procedures in the implementation, it is useful if the generator systems have a minimal number of elements and also that the coefficients of (a permutation of) the generators can form a triangular matrix.

Definition 3.18 (Generator System in Minimal Form.) *Suppose $\mathcal{G} = (L, Q, P)$ is a generator system in $\mathbb{Q}^n$. Then we say that $\mathcal{G}$ is in* minimal form *if either $L = Q = P = \varnothing$ or $\#\,P = 1$ and, for each generator $\mathbf{v} \in L \cup Q$, the following hold:*

1. *if $\mathrm{piv}_>(\mathbf{v}) = k$, then $v_k > 0$;*
2. *for all $\mathbf{v}' \in (L \cup Q) \setminus \{\mathbf{v}\}$, $\mathrm{piv}_>(\mathbf{v}') \neq \mathrm{piv}_>(\mathbf{v})$.*

We will now show how to produce an n-dimensional grid in minimal form given that it is represented by a generator system which consists of m generators.

Proposition 3.19 *There exists an algorithm that, for each generator system $\mathcal{G}$ in $\mathbb{Q}^n$, computes a generator system $\mathcal{G}'$ in minimal form such that $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G})$. Letting $\mathcal{G} = (L, Q, P)$ and $m := \#\,L + \#\,Q + \#\,P$, the algorithm has worst-case complexity $\mathrm{O}\big(mn\min\{m, n\}\big)$.*

Proof. If $P = \varnothing$, then $\mathrm{ggen}(\mathcal{G}) = \varnothing$; in this case, let $\mathcal{G}' = (\varnothing, \varnothing, \varnothing)$. Suppose now that there exists a point $\mathbf{p} \in P$. Let $\mathcal{G}_\mathbf{p} = \big(L, Q_\mathbf{p}, \{\mathbf{p}\}\big)$, where

$$Q_\mathbf{p} = \Big(Q \cup \big\{\, \mathbf{p}'' - \mathbf{p} \in \mathbb{Q}^n \;\big|\; \mathbf{p}'' \in P \setminus \{\mathbf{p}\} \,\big\}\Big) \setminus L.$$

Since $m = \#\,L + \#\,Q + \#\,P$, we obtain $\#\,L + \#\,Q_\mathbf{p} < m$. Then $\mathrm{ggen}(\mathcal{G}_\mathbf{p}) = \mathrm{ggen}(\mathcal{G})$ since

$$\mathrm{linear.hull}(L) + \mathrm{int.hull}(Q) + \mathrm{int.affine.hull}(P) = \\ \mathrm{linear.hull}(L) + \mathrm{int.hull}(Q_\mathbf{p}) + \mathrm{int.affine.hull}\big(\{\mathbf{p}\}\big).$$

To prove the result, we first define the key transformation step in the algorithm and show that the resulting generator system describes the same grid. Suppose there exist distinct generators

$\mathbf{v}_1, \mathbf{v}_2 \in L \cup Q_{\mathbf{p}}$ such that $\mathrm{piv}_>(\mathbf{v}_1) = \mathrm{piv}_>(\mathbf{v}_2) = i \leq n$. We will define a generator system $\mathcal{G}'' = \big(L'', Q''_{\mathbf{p}}, \{\mathbf{p}\}\big)$ in $\mathbb{Q}^n$ where $L'' \cup Q''_{\mathbf{p}} = \big(L \cup Q_{\mathbf{p}} \setminus \{\mathbf{v}_1, \mathbf{v}_2\}\big) \cup \big(\{\mathbf{v}''_1, \mathbf{v}''_2\} \setminus \{\mathbf{0}\}\big)$, $\mathrm{ggen}(\mathcal{G}'') = \mathrm{ggen}(\mathcal{G}_{\mathbf{p}})$, $\mathrm{piv}_>(\mathbf{v}''_1) = i$ and, if $\mathbf{v}''_2 \neq \mathbf{0}$, $\mathrm{piv}_>(\mathbf{v}''_2) > i$. There are three cases.

1. Suppose that $\{\mathbf{v}_1, \mathbf{v}_2\} \subseteq L$. Then, using Gaussian elimination, let

$$\begin{aligned} &\mathbf{v}''_1 = \mathbf{v}_1, && \mathbf{v}''_2 = \mathbf{v}_2 - (v_{2i}/v_{1i})\mathbf{v}_1; \\ &L'' = \big(L \setminus \{\mathbf{v}_2\}\big) \cup \big(\{\mathbf{v}''_2\} \setminus \{\mathbf{0}\}\big), && Q''_{\mathbf{p}} = Q_{\mathbf{p}}. \end{aligned}$$

2. Suppose that $\mathbf{v}_1 \in L$ and $\mathbf{v}_2 \in Q_{\mathbf{p}}$ or vice-versa; without loss of generality, we assume that $\mathbf{v}_1 \in L$. Then, using Gaussian elimination, let

$$\begin{aligned} &\mathbf{v}''_1 = \mathbf{v}_1, && \mathbf{v}''_2 = \mathbf{v}_2 - (v_{2i}/v_{1i})\mathbf{v}_1; \\ &L'' = L, && Q''_{\mathbf{p}} = \big(Q_{\mathbf{p}} \setminus \{\mathbf{v}_2\}\big) \cup \big(\{\mathbf{v}''_2\} \setminus \{\mathbf{0}\}\big). \end{aligned}$$

3. Suppose that $\{\mathbf{v}_1, \mathbf{v}_2\} \subseteq Q_{\mathbf{p}}$. Let $\mathrm{gcdext}(v_{1i}, v_{2i}) = \big(r, (s, t)\big)$,

$$\begin{aligned} &\mathbf{v}''_1 = s\mathbf{v}_1 + t\mathbf{v}_2, && \mathbf{v}''_2 = (-v_{2i}/r)\mathbf{v}_1 + (v_{1i}/r)\mathbf{v}_2; \\ &L'' = L, && Q''_{\mathbf{p}} = \big(Q_{\mathbf{p}} \setminus \{\mathbf{v}_1, \mathbf{v}_2\}\big) \cup \big(\{\mathbf{v}''_1, \mathbf{v}''_2\} \setminus \{\mathbf{0}\}\big). \end{aligned}$$

In all cases, $\mathrm{ggen}(\mathcal{G}'') = \mathrm{ggen}(\mathcal{G}_{\mathbf{p}})$. Note that these transformations require a computation for each coefficient of the considered generators so that their complexity is $\mathrm{O}\big(n\big)$.

The proof of the result is by induction on $n + 1 - i$, where

$$i := \min\Big(\{n+1\} \cup \big\{\, j \in \mathbb{N} \;\big|\; \exists \mathbf{v}_1 \neq \mathbf{v}_2 \in L \cup Q_{\mathbf{p}} \,.\, j = \mathrm{piv}_>(\mathbf{v}_1) = \mathrm{piv}_>(\mathbf{v}_2) \,\big\}\Big).$$

The base case is when $i = n+1$, in which case $\mathcal{G}_{\mathbf{p}}$ is already in minimal form, so let $\mathcal{G}' = \mathcal{G}_{\mathbf{p}}$. For the step case, we apply the transformations (1), (2) and (3) until no more transformations are applicable with index i; that is when we obtain a generator system $\mathcal{G}_j = \big(L_j, Q_j, \{\mathbf{p}\}\big)$ for which $j > i$ is the least value such that, if there exists a pair of distinct generators $\mathbf{v}_1, \mathbf{v}_2 \in L_j \cup Q_j$, then $j = \mathrm{piv}_>(\mathbf{v}_1) = \mathrm{piv}_>(\mathbf{v}_2)$; $j = n + 1$ if such a pair does not exist. We note that we will have to perform these transformations at most $m - 1$ times for each step, where $\# L + \# Q_{\mathbf{p}} = m - 1$, so that the complexity of each step is $\mathrm{O}\big(nm\big)$. By the inductive hypothesis, we can compute $\mathcal{G}'$ in minimal form such that $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G}_j)$. Therefore $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G})$. As we can iterate at most $\min\{m, n\}$ times over the step case, the algorithm has complexity $\mathrm{O}\big(mn \min\{m, n\}\big)$. $\square$

As for Proposition 3.17, the algorithm mentioned in Proposition 3.19 is based on the Hermite normal form algorithm [67, 76]. Note also that, when $m < n$, the complexity of this algorithm is just $\mathrm{O}\big(m^2 n\big)$. If the congruence system $\mathcal{C}$ (or generator system $\mathcal{G}$) is for a rectilinear grid then the complexity of computing the minimal form is at worst $\mathrm{O}\big(m \min\{m, n\}\big)$. Note that the

congruence system $\mathcal{C}$ (resp., generator system $\mathcal{G}$) for a non-empty grid is in minimal form if and only if the homogeneous form $\hat{\mathcal{C}}$ for $\mathcal{C}$ (resp., $\hat{\mathcal{G}}$ for $\mathcal{G}$) is in minimal form.

We will now show how we can produce a minimal form which can achieve a canonical form for a congruence system by producing an algorithm which takes a system in minimal form and returns an equivalent system which represents the same grid and whose coefficients are all as small as possible in absolute value. We will first require the definition of pivot equivalence to be extended to consider a set of congruences.

Definition 3.20 (Pivot Equivalent Congruence Systems.) *Congruence systems in minimal form $\mathcal{C}_1$ and $\mathcal{C}_2$ are said to be* pivot equivalent *if: for each $\beta \in \mathcal{C}_1$, there exists a $\gamma \in \mathcal{C}_2$ such that $\beta \Uparrow \gamma$; for each $\gamma \in \mathcal{C}_2$, there exists a $\beta \in \mathcal{C}_1$ such that $\gamma \Uparrow \beta$.*

Definition 3.21 (Congruence System in Strong Minimal Form.) *A congruence system $\mathcal{C}$ in $\mathbb{Q}^n$ is in* strong minimal form *if $\mathcal{C}$ is in minimal form and, for each pair of distinct proper congruences*

$$\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_1 b\big),\ \gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_1 d\big) \in \mathcal{C},$$

if $\mathrm{piv}_<(\gamma) = k > 0$, then $-c_k < 2a_k \leq c_k$.

A congruence system in minimal form can always be reduced to a congruence system in strong minimal form that describes the same grid.

Proposition 3.22 *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$ in minimal form. Then there exists an algorithm with complexity $\mathrm{O}\big(n^3\big)$ for converting $\mathcal{C}$ to a congruence system $\mathcal{C}'$ in strong minimal form such that $\mathcal{C}$ is pivot equivalent to $\mathcal{C}'$ and $\mathrm{gcon}(\mathcal{C}) = \mathrm{gcon}(\mathcal{C}')$.*

Proof. Suppose that $\mathcal{C}$ is not in strong minimal form. Then, by Definition 3.21, there exists a proper congruence $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_1 b\big) \in \mathcal{C}$, such that the following holds:

1. there exists $i > 0$ and a proper congruence $\gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_1 d\big) \in \mathcal{C}\backslash\{\beta\}$ where $\mathrm{piv}_<(\gamma) = i$ and either $2a_i \leq -c_i$ or $2a_i > c_i$.

Suppose that $0 \leq k \leq n$ is the maximum value for the index i such that condition (1) holds.

We show, by induction on k, that there exists a sequence of at most n transformations, each of which having complexity $\mathrm{O}\big(n\big)$, from β to the congruence $\beta' = \big(\langle \mathbf{a}', \mathbf{x}\rangle \equiv_1 b'\big)$, such that, if $\mathcal{C}' := \big(\mathcal{C} \setminus \{\beta\}\big) \cup \{\beta'\}$, then $\mathrm{gcon}(\mathcal{C}') = \mathrm{gcon}(\mathcal{C})$ and condition (1) (when β is replaced by β') does not hold.

If $k = 0$, then condition (1) does not hold for β. Therefore let $\beta' = \beta$.

Suppose now that $k > 0$ so that condition (1) holds for $i = k$. As $\mathcal{C}$ is in minimal form, $k < \mathrm{piv}_<(\mathbf{a})$. Let

$$\mathbf{a}'' = \begin{cases} \mathbf{a} - \left\lceil \frac{a_k}{c_k} \right\rfloor \mathbf{c} \quad \text{and} \quad b'' = b - \left\lceil \frac{a_k}{c_k} \right\rfloor d, & \text{if } a_k \bmod c_k > \frac{a_k}{2}; \\ \mathbf{a} - \left\lfloor \frac{a_k}{c_k} \right\rfloor \mathbf{c} \quad \text{and} \quad b'' = b - \left\lfloor \frac{a_k}{c_k} \right\rfloor d, & \text{if } a_k \bmod c_k \leq \frac{a_k}{2}. \end{cases}$$

Then $-c_k < 2a''_k \leq c_k$. Also, for $k+1 \leq j \leq n$, we have $c_j = 0$ so that $a_j = a''_j$ and $\mathrm{piv}_<(\mathbf{a}'') = \mathrm{piv}_<(\mathbf{a})$. Letting $\beta'' := \big(\langle \mathbf{a}'', \mathbf{x}\rangle \equiv_1 b''\big)$ and $\mathcal{C}'' := \big(\mathcal{C} \setminus \{\beta\}\big) \cup \{\beta''\}$, we have $\mathrm{gcon}(\mathcal{C}'') = \mathrm{gcon}(\mathcal{C})$. Note that this transformation has a complexity $\mathrm{O}\big(n\big)$. As k'', the maximum index such that condition (1) holds for β'', is strictly less than k we can apply the inductive hypothesis to $\mathcal{C}''$ and β''. Thus there is a sequence of at most $n-1$ transformations from β'' to β' such that, $\mathrm{gcon}\Big(\big(\mathcal{C}'' \setminus \{\beta''\}\big) \cup \{\beta'\}\Big) = \mathrm{gcon}(\mathcal{C}'')$ and condition (1) (when β is replaced by β') does not hold. Thus there is a sequence of at most n transformations from β to β' such that $\mathrm{gcon}\Big(\big(\mathcal{C} \setminus \{\beta\}\big) \cup \{\beta'\}\Big) = \mathrm{gcon}(\mathcal{C})$. As each of the individual steps has complexity $\mathrm{O}\big(n\big)$, the sequence of transformations has complexity $\mathrm{O}\big(n^2\big)$.

We repeat this sequence of transformations for each proper congruence in $\mathcal{C}$ to obtain a congruence system $\mathcal{C}'$ such that, for each proper congruence $\beta' \in \mathcal{C}'$, condition (1) does not hold. Thus, by Definition 3.21, $\mathcal{C}'$ is in strong normal form. Thus, as there are at most n proper congruences in $\mathcal{C}$ since, by hypothesis, $\mathcal{C}$ is in minimal form, the complexity of computing the strong minimal form is $\mathrm{O}\big(n^3\big)$. $\square$

Note that if the congruence system is in homogeneous form them the strong minimal form algorithm will also reduce all the former inhomogeneous terms to be as small as possible, thus the set of proper congruences will be in canonical form.

As for the congruence system we can also extend the notion of pivot equivalence to consider generator systems.

Definition 3.23 (Pivot Equivalent Generator Systems.) *We say that generator systems $\mathcal{G}_1 = \big(L, Q, \{\mathbf{p}\}\big)$ and $\mathcal{G}_2 = \big(L', Q', \{\mathbf{p}'\}\big)$ in minimal form are* pivot equivalent *if: for each $\mathbf{q} \in Q$, there exists $\mathbf{q}' \in Q'$ such that $\mathbf{q} \Downarrow \mathbf{q}'$, and, for each $\boldsymbol{\ell} \in L$, there exists $\boldsymbol{\ell}' \in L'$ such that $\mathrm{piv}_>(\boldsymbol{\ell}) = \mathrm{piv}_>(\boldsymbol{\ell}')$; for each $\mathbf{q}' \in Q'$, there exists $\mathbf{q} \in Q$ such that $\mathbf{q}' \Downarrow \mathbf{q}$, and, for each $\boldsymbol{\ell}' \in L'$, there exists $\boldsymbol{\ell} \in L$ such that $\mathrm{piv}_>(\boldsymbol{\ell}') = \mathrm{piv}_>(\boldsymbol{\ell})$.*

We can also define the notion of strong minimal form for a Generator system.

Definition 3.24 (Generator System in Strong Minimal Form.) *A generator system $\mathcal{G}$ in $\mathbb{Q}^n$, where $\mathcal{G} = \big(L, Q, \{\mathbf{p}\}\big)$ is in* strong minimal form *if $\mathcal{G}$ is in minimal form and, for each pair of distinct vectors $\mathbf{u}, \mathbf{v} \in Q$, if $\mathrm{piv}_>(\mathbf{v}) = k \leq n$, then $-v_k < 2u_k \leq v_k$.*

A generator system in minimal form can always be reduced to a generator system in strong minimal form that describes the same grid.

Proposition 3.25 *Let $\mathcal{G}$ be a generator system in $\mathbb{Q}^n$ in minimal form. Then there exists an algorithm with complexity $\mathrm{O}\big(n^3\big)$ for converting $\mathcal{G}$ to a generator system $\mathcal{G}'$ in strong minimal form such that $\mathcal{G}$ is pivot equivalent to $\mathcal{G}'$ and $\mathrm{ggen}(\mathcal{G}) = \mathrm{ggen}(\mathcal{G}')$.*

Proof. Suppose that $\mathcal{G} = \big(L, Q, \{\mathbf{p}\}\big)$ is not in strong minimal form. Then, by Definition 3.24, then there exists a generator $\mathbf{u} \in Q$, such that the following holds:

1. there exists $1 \leq i \leq n$ and a generator $\mathbf{v} \in Q \setminus \{\mathbf{u}\}$ where $\mathrm{piv}_{>}(\mathbf{v}) = i$ and either $2u_i \leq -v_i$ or $2u_i > v_i$.

If condition (1) does not hold for $\mathbf{u}$; let $\mathbf{u}' = \mathbf{u}$.

Suppose now that condition (1) holds and that $k \in \{1, \ldots, n\}$ is such that $n + 1 - k$ is the minimum value for the index i for which this condition holds.

We show, by induction on k, that there exists a sequence of at most n transformations, each of which having complexity $\mathrm{O}\big(n\big)$, from $\mathbf{u}$ to the generator $\mathbf{u}'$, such that, if $\mathcal{G}' := \big(\mathcal{G} \setminus \{\mathbf{u}\}\big) \cup \{\mathbf{u}'\}$, then $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G})$ and condition (1) (when $\mathbf{u}$ is replaced by $\mathbf{u}'$) does not hold.

As $\mathcal{G}$ is in minimal form, $i > \mathrm{piv}_{>}(\mathbf{u})$. Let

$$\mathbf{u}'' = \begin{cases} \mathbf{u} - \left\lceil \frac{u_i}{v_i} \right\rceil \mathbf{v}, & \text{if } u_i \bmod v_i > \frac{v_i}{2}; \\ \mathbf{u} - \left\lfloor \frac{u_i}{v_i} \right\rfloor \mathbf{v}, & \text{if } u_i \bmod v_i \leq \frac{v_i}{2}. \end{cases}$$

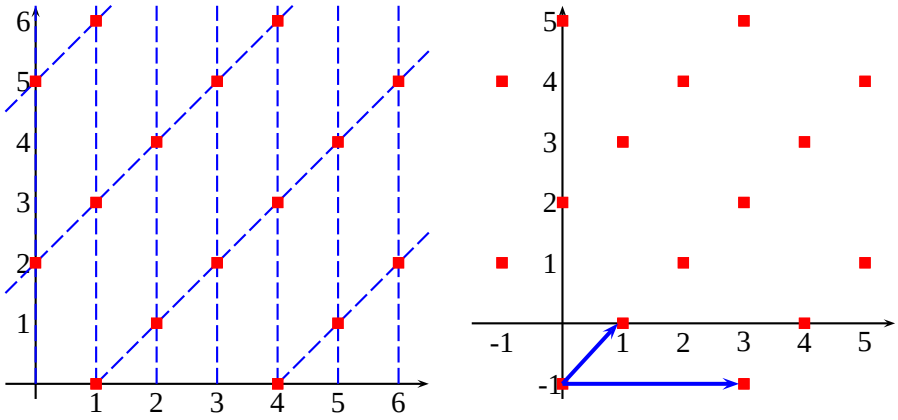
Then $-v_i < 2u_i'' \leq v_i$. Also, for $1 \leq j \leq i - 1$, we have $v_j = 0$ so that $u_j = u_j''$ and $\mathrm{piv}_{>}(\mathbf{u}'') = \mathrm{piv}_{>}(\mathbf{u})$. Letting $\mathcal{G}'' := \big(\mathcal{G} \setminus \{\mathbf{u}\}\big) \cup \{\mathbf{u}''\}$, we have $\mathrm{ggen}(\mathcal{G}'') = \mathrm{ggen}(\mathcal{G})$. Note that this transformation has a complexity $\mathrm{O}\big(n\big)$. As $n + 1 - k''$, the minimum index such that condition (1) holds for $\mathbf{u}''$, is strictly greater than $n + 1 - k$, we have that k'' is strictly less than k, therefore we can apply the inductive hypothesis to $\mathcal{G}''$ and $\mathbf{u}''$. Thus there is a sequence of at most $n - 1$ transformations from $\mathbf{u}''$ to $\mathbf{u}'$ such that, $\mathrm{ggen}\Big(\big(\mathcal{G}'' \setminus \{\mathbf{u}''\}\big) \cup \{\mathbf{u}'\}\Big) = \mathrm{ggen}(\mathcal{G}'')$ and condition (1) (when $\mathbf{u}$ is replaced by $\mathbf{u}'$) does not hold. Thus there is a sequence of at most n transformations from $\mathbf{u}$ to $\mathbf{u}'$ such that $\mathrm{ggen}\Big(\big(\mathcal{G} \setminus \{\mathbf{u}\}\big) \cup \{\mathbf{u}'\}\Big) = \mathrm{ggen}(\mathcal{G})$. As each of the individual steps has complexity $\mathrm{O}\big(n\big)$, the sequence of transformations has complexity $\mathrm{O}\big(n^2\big)$.

We repeat this sequence of transformations for each parameter in $\mathcal{G}$ to obtain a generator system $\mathcal{G}'$ such that, for each parameter $\mathbf{u}' \in \mathcal{G}'$, condition (1) does not hold. Thus, by Definition 3.24, $\mathcal{G}'$ is in strong normal form. Thus, as there are at most n parameters in $\mathcal{G}$ since, by hypothesis, $\mathcal{G}$ is in minimal form, the complexity of computing the strong minimal form is $\mathrm{O}\big(n^3\big)$. $\square$

Note that if a generator system $\mathcal{G} = \big(L, Q, P\big)$ is in homogeneous form, $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q} \cup \hat{P}, \{\mathbf{0}\}\big)$, when the strong minimal form algorithm is applied, $\hat{\mathcal{G}}$ will be reduced so that all the coefficients of the former inhomogeneous term P will be as small as possible as well as Q, thus the set of parameters and point will be in canonical form. If the congruence system $\mathcal{C}$ (resp., generator system $\mathcal{G}$) is for a rectilinear grid then the congruence system $\mathcal{C}$ (resp., generator system $\mathcal{G}$) is already is strong minimal form.

Example 3.26 *Consider the grid* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$ *where* $\mathcal{C} := \{x \equiv_1 0,\ x - y \equiv_3 1\}$ *and*

$$\mathcal{G} := \left(\varnothing, \begin{pmatrix} 1 & 0 \\ 1 & 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right).$$



(a) Grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$. (b) Grid $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$.

Figure 3.5: A grid in $\mathbb{R}^2$ represented by systems in strong minimal form.

$\mathcal{L} = \mathrm{gcon}(\mathcal{C})$ can be seen in Figure 3.5(a) and $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$ can be seen in Figure 3.5(b). Let $\hat{\mathcal{L}}$ be the homogeneous form of $\mathcal{L}$. Then the matrix forms for $\hat{\mathcal{C}}$ and $\hat{\mathcal{G}}$ in strong minimal form are given by

$$\hat{\mathcal{C}} := \left(\begin{pmatrix} 3 & 0 & 1 \\ 0 & 3 & -1 \\ 0 & 0 & 1 \end{pmatrix}, \varnothing \right)$$

and

$$\hat{\mathcal{G}} := \left(\varnothing, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right).$$

Now that we have defined the two descriptions for the grid we will now show that an algorithm exists that can transfer a grid described by one representation to an equivalent grid described by the other representation.

By considering the matrix forms of the representations which are in minimal homogeneous forms, we can build the conversion algorithms using those for matrix inversion. Informally this is appropriate since suppose that the generator system $\hat{\mathcal{G}} = (\varnothing, \hat{Q}, \{\mathbf{0}\})$ in $\mathbb{Q}^{n+1}$ is in minimal homogeneous form and $\hat{Q}$ is a non-singular square matrix. Letting $\hat{\mathcal{L}} = \mathrm{ggen}(\hat{\mathcal{G}}) = \{ \hat{Q}\pi \in \mathbb{Q}^{n+1} \mid \pi \in \mathbb{Z}^n \}$, then we also have $\hat{\mathcal{L}} = \{ \hat{\mathbf{v}} \in \mathbb{Q}^{n+1} \mid \hat{Q}^{-1}\hat{\mathbf{v}} \equiv_1 0 \}$. So $(\hat{Q}^{-1}, \varnothing)$ is the matrix form of a congruence system in minimal homogeneous form that represents the same grid $\hat{\mathcal{L}}$. Similarly we can use matrix inversion to convert the matrix form of a homogeneous congruence system in minimal form consisting of $n+1$ proper congruences for a grid $\hat{\mathcal{L}}$ to a generator system in minimal homogeneous form for $\hat{\mathcal{L}}$. When the matrices to be inverted have less than $n+1$ linearly independent columns, the algorithms we propose first add vectors $\hat{\mathbf{e}}_i$ where $1 \leq i \leq n$, as necessary, so as to make the matrices non-singular and hence invertible. For example, suppose that the generator system $\hat{\mathcal{G}} = (\varnothing, \hat{Q}, \{\mathbf{0}\})$ in $\mathbb{Q}^{n+1}$ is such that for all $\mathbf{q} \in \hat{Q}$, $\mathrm{piv}_{>}(\mathbf{q}) \neq i$.

Then $\hat{\mathbf{e}}_i$ is added to the generator system.

Lemma 3.27 *Let $\hat{L}, \hat{Q}, \hat{M}, \hat{F}, \hat{E}, \hat{N}$ be matrices in $\mathbb{Q}^{n+1}$ such that: $\#\,\hat{N} = \#\,\hat{L}$, $\#\,\hat{F} = \#\,\hat{Q} > 0$ and $\#\,\hat{E} = \#\,\hat{M}$. Also let $(\hat{L}, \hat{Q}, \hat{M})$ and $(\hat{N}, \hat{F}, \hat{E})$ be square and non-singular matrices where $(\hat{N}, \hat{F}, \hat{E})^{\mathrm{T}} = (\hat{L}, \hat{Q}, \hat{M})^{-1}$. Suppose $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ is a generator system in minimal homogeneous form in $\mathbb{Q}^{n+1}$ (resp., $(\hat{F}, \hat{E})$ is the matrix form of a congruence system $\hat{\mathcal{C}}$ in minimal homogeneous form) and $\hat{M}$ (resp., $\hat{N}$) a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\hat{\mathbf{e}}_i$. Then $(\hat{F}, \hat{E})$ is the matrix form of a congruence system $\hat{\mathcal{C}}$ in minimal homogeneous form (resp., $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ is a generator system in minimal homogeneous form), $\hat{N}$ (resp., $\hat{M}$) is a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\hat{\mathbf{e}}_i$ and*

1. *$\#\,\hat{Q} = \#\,\hat{F} = n + 1 - \#\,\hat{L} - \#\,\hat{E} > 0$;*
2. *$\mathrm{gcon}(\hat{\mathcal{C}}) = \mathrm{ggen}(\hat{\mathcal{G}})$;*
3. *there exists $\hat{\mathbf{q}} \in \hat{Q}$ if and only if there exists $\hat{\mathbf{a}} \in \hat{F}$, such that, for some $k \in \{1, \ldots, n\}$, $\mathrm{piv}_{>}(\hat{\mathbf{q}}) = \mathrm{piv}_{<}(\hat{\mathbf{a}}) = k$ and $q_k a_k = 1$;*
4. *for all $\hat{\boldsymbol{\ell}} \in \hat{L}$ and $\hat{\mathbf{a}} \in \hat{E}$, $\mathrm{piv}_{>}(\hat{\boldsymbol{\ell}}) \neq \mathrm{piv}_{<}(\hat{\mathbf{a}})$.*

Proof. Suppose $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ is a generator system in minimal homogeneous form in $\mathbb{Q}^{n+1}$ and $\hat{M}$ a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\hat{\mathbf{e}}_i$. By the hypothesis on the cardinalities of the matrices, (1) holds. By Definition 3.18, $(\hat{L}, \hat{Q}, \hat{M})$ is a permutation of a matrix in lower triangular form where the diagonal elements are all positive and there exists $\hat{\mathbf{p}} \in \hat{Q}$ such that $p_0 = 1$. Also, by hypothesis,

$$(\hat{N}, \hat{F}, \hat{E})^{\mathrm{T}} = (\hat{L}, \hat{Q}, \hat{M})^{-1} \tag{3.5}$$

so that $(\hat{N}, \hat{F}, \hat{E})^{\mathrm{T}}$ is a permutation of a matrix in upper triangular form where the diagonal elements are all positive. Hence, by Definition 3.15, $\hat{\mathcal{C}}$ is also in minimal form. Also, by the hypothesis on the cardinalities of the matrices, (3) and (4) follow from (3.5).

Since $(\hat{L}, \hat{Q}, \hat{M})$ is in lower triangular form and $\hat{\mathbf{p}} \in \hat{Q}$ such that $p_0 = 1$, the first row of the matrix $(\hat{L}, \hat{Q}, \hat{M})$ is of the form $\hat{\mathbf{e}}_i$ where $i \in \{0, \ldots, n\}$ is the index of $\hat{\mathbf{p}}$ in $(\hat{L}, \hat{Q}, \hat{M})$. Thus the i-th vector in $(\hat{N}, \hat{F}, \hat{E})$ must be in $\hat{F}$ and have the form $\hat{\mathbf{e}}_0$. It follows that $\hat{\mathcal{C}}$ is in homogeneous form.

Finally, using (3.5) and letting $\#\,\hat{L} = \ell$ and $\#\,\hat{Q} = q + 1$, (2) holds since we have

$$\begin{aligned}
\hat{\mathbf{x}} \in \mathrm{ggen}(\hat{\mathcal{G}}) &\iff \hat{\mathbf{x}} = \hat{L}\lambda + \hat{Q}\pi + \hat{M}\mathbf{0}, && \text{for } \lambda \in \mathbb{R}^{\ell},\ \pi \in \mathbb{Z}^{q+1} \\
&\iff \hat{\mathbf{x}} = (\hat{L}, \hat{Q}, \hat{M})(\lambda^{\mathrm{T}}, \pi^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}, && \text{for } \lambda \in \mathbb{R}^{\ell},\ \pi \in \mathbb{Z}^{q+1} \\
&\iff (\hat{L}, \hat{Q}, \hat{M})^{-1}\hat{\mathbf{x}} = (\lambda^{\mathrm{T}}, \pi^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}, && \text{for } \lambda \in \mathbb{R}^{\ell},\ \pi \in \mathbb{Z}^{q+1} \\
&\iff (\hat{N}, \hat{F}, \hat{E})^{\mathrm{T}}\hat{\mathbf{x}} = (\lambda^{\mathrm{T}}, \pi^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}, && \text{for } \lambda \in \mathbb{R}^{\ell},\ \pi \in \mathbb{Z}^{q+1} \\
&\iff \hat{N}^{\mathrm{T}}\hat{\mathbf{x}} = \lambda,\ \hat{F}^{\mathrm{T}}\hat{\mathbf{x}} = \pi,\ \hat{E}^{\mathrm{T}}\hat{\mathbf{x}} = \mathbf{0}, && \text{for } \lambda \in \mathbb{R}^{\ell},\ \pi \in \mathbb{Z}^{q+1} \\
&\iff \hat{F}^{\mathrm{T}}\hat{\mathbf{x}} \equiv_1 \mathbf{0},\ \hat{E}^{\mathrm{T}}\hat{\mathbf{x}} = \mathbf{0} \\
&\iff \hat{\mathbf{x}} \in \mathrm{gcon}(\hat{\mathcal{C}}).
\end{aligned}$$

The proof when it is assumed that $(\hat{F}, \hat{E})$ is the matrix form of a congruence system $\hat{\mathcal{C}}$ in minimal homogeneous form is similar. □

Lemma 3.28 *There exists a computable, invertible function that converts a generator system $\mathcal{G} = \big(L, Q, \{\mathbf{p}\}\big)$ in $\mathbb{Q}^n$ in minimal form to a consistent congruence system $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$ in $\mathbb{Q}^n$ in minimal form where $\mathcal{E}$ are equalities and $\mathcal{F}$ are proper congruences and such that*

5. $\# Q = \# \mathcal{F} = n - \# L - \# \mathcal{E}$;
6. $\mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$;
7. *there exists $\mathbf{q} \in Q$ if and only if there exists $\beta = \big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_1 0\big) \in \mathcal{F}$, such that, for some $k \in \{1, \ldots, n\}$, $\mathrm{piv}_>(\mathbf{q}) = \mathrm{piv}_<(\mathbf{a}) = k$ and $q_k a_k = 1$;*
8. *for all $\boldsymbol{\ell} \in L$ and $\beta \in \mathcal{E}$, $\mathrm{piv}_>(\boldsymbol{\ell}) \neq \mathrm{piv}_<(\beta)$.*

Proof. Let $\# L = \ell$, $\# Q = q$ and $\hat{\mathcal{G}} := \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ where

$$\begin{aligned} \hat{\mathbf{p}} &:= (1, \mathbf{p}^{\mathrm{T}})^{\mathrm{T}}, \\ \hat{L} &:= \big\{\, (0, \boldsymbol{\ell}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \boldsymbol{\ell} \in L \,\big\}, \\ \hat{Q} &:= \big\{\, (0, \mathbf{q}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \mathbf{q} \in Q \,\big\} \cup \{\hat{\mathbf{p}}\}. \end{aligned} \tag{3.6}$$

Then $\hat{\mathcal{G}}$ is the homogeneous form for $\mathcal{G}$, $\# \hat{Q} > 0$ and, as $\mathcal{G}$ is in minimal form, $\hat{\mathcal{G}}$ is also in minimal form. By Lemma 3.27, there exists a computable invertible function that will convert $\hat{\mathcal{G}}$ in $\mathbb{Q}^{n+1}$ to a congruence system $\hat{\mathcal{C}} = \hat{\mathcal{F}} \cup \hat{\mathcal{E}}$ in $\mathbb{Q}^{n+1}$ in minimal homogeneous form where $\hat{\mathcal{F}}$ are proper congruences and $\hat{\mathcal{E}}$ are equalities and such that properties (1), (2), (3), and (4) in Lemma 3.27 hold. It follows that $\hat{\mathcal{C}}$ is the homogeneous form for a congruence system $\mathcal{C} = \mathcal{F} \cup \mathcal{E}$ in minimal form, where

$$\begin{aligned} \mathcal{E} &= \Big\{ \big(\langle \mathbf{a}, \mathbf{x} \rangle = b\big) \;\Big|\; \big\langle (-b, \mathbf{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\mathbf{x}} \big\rangle = 0 \in \hat{\mathcal{E}} \Big\}, \\ \mathcal{F} &= \Big\{ \big(\langle \mathbf{a}, \mathbf{x} \rangle \equiv_1 b\big) \;\Big|\; \big\langle (-b, \mathbf{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\mathbf{x}} \big\rangle \equiv_1 0 \in \hat{\mathcal{F}} \setminus \{\hat{\mathbf{e}}_0\} \Big\} \end{aligned}$$

and that properties (5), (6), (7) and (8) for $\mathcal{C}$ and $\mathcal{G}$ hold. □

The following proposition shows how to convert a congruence system into a generator system which describes the same grid.

Proposition 3.29 *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$ in minimal form for a non-empty grid; $(\hat{F}, \hat{E})$ the matrix form of the homogeneous form for $\mathcal{C}$; $\hat{N}$ a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\hat{\mathbf{e}}_i$, with $i \in \{0, \ldots, n\}$, and such that $(\hat{N}, \hat{F}, \hat{E})$ is square and non-singular; and $(\hat{L}, \hat{Q}, \hat{M}) := \big((\hat{N}, \hat{F}, \hat{E})^{-1}\big)^{\mathrm{T}}$ where $\# \hat{L} = \# \hat{N}$, $\# \hat{Q} = \# \hat{F}$ and $\# \hat{M} = \# \hat{E}$. Then $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ is the homogeneous form for a generator system $\mathcal{G}$ in minimal form and $\mathrm{ggen}(\mathcal{G}) = \mathrm{gcon}(\mathcal{C})$.*

Proof. By the hypothesis and Lemma 3.27, $\hat{\mathcal{G}} = (\hat{L}, \hat{Q}, \{\mathbf{0}\})$ is a generator system in minimal homogeneous form and, by property (2) of Lemma 3.27, $\mathrm{gcon}(\hat{\mathcal{C}}) = \mathrm{ggen}(\hat{\mathcal{G}})$. Therefore, $\hat{\mathcal{G}}$ is the homogeneous form for $\mathcal{G}$, a generator system in minimal form, $\hat{\mathcal{C}}$ is the homogeneous form for $\mathcal{C}$, a congruence system in minimal form, and $\mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$. □

The following proposition shows how to convert a generator system into a congruence system which describes the same grid.

Proposition 3.30 *Let $\mathcal{G}$ be a generator system in $\mathbb{Q}^n$ in minimal form for a non-empty grid; $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ the homogeneous form for $\mathcal{G}$; $\hat{M}$ a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\hat{\mathbf{e}}_i$, with $i \in \{0, \ldots, n\}$, and such that $(\hat{L}, \hat{Q}, \hat{M})$ is square and non-singular; and $(\hat{N}, \hat{F}, \hat{E}) := \big((\hat{L}, \hat{Q}, \hat{M})^{-1}\big)^{\mathrm{T}}$ where $\#\,\hat{N} = \#\,\hat{L}$, $\#\,\hat{F} = \#\,\hat{Q}$ and $\#\,\hat{E} = \#\,\hat{M}$. Then $(\hat{F}, \hat{E})$ is the matrix form of the homogeneous form for a congruence system $\mathcal{C}$ in minimal form and $\mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$.*

Proof. By the hypothesis and Lemma 3.27, $(\hat{F}, \hat{E})$ is the matrix form of a congruence system $\hat{\mathcal{C}}$ in minimal homogeneous form and, by property (2) of Lemma 3.27, $\mathrm{gcon}(\hat{\mathcal{C}}) = \mathrm{ggen}(\hat{\mathcal{G}})$. Therefore, $\hat{\mathcal{C}}$ is the homogeneous form for $\mathcal{C}$, a congruence system in minimal form, $\hat{\mathcal{G}}$ is the homogeneous form for $\mathcal{G}$, a generator system in minimal form, and $\mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$. □

Both of the algorithms described for conversion between the two systems just perform matrix inversion; so their complexity depends on the inversion algorithm adopted in the implementation. As far as we know, the current best theoretical worst-case complexity is $\mathrm{O}\big(n^{2.376}\big)$ [23]. Note that, in the current implementation in the PPL, the conversion algorithm is based on the Gaussian elimination method, which has complexity $\mathrm{O}\big(n^3\big)$. If however the congruence system $\mathcal{C}$ (or generator system $\mathcal{G}$) is for a rectilinear grid then the complexity of the conversion algorithm is just $\mathrm{O}\big(n\big)$.

The following example will show that the conversion algorithm does not respect strong minimal form of the given system.

Example 3.31 *Suppose we have the grid $\mathcal{L}$ which is in strong minimal form and homogeneous form. Let $\mathcal{L} = \mathrm{ggen}(\varnothing, \hat{Q}, \{\mathbf{0}\})$ where*

$$\hat{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ -1 & 2 & 4 & 0 \\ 2 & -1 & 2 & 4 \end{pmatrix}.$$

Then after applying the conversion algorithm to the matrix $\hat{Q}$, we get the matrix $\hat{F}$ such that

$$\hat{F} = \begin{pmatrix} 8 & -4 & 4 & -7 \\ 0 & 2 & -1 & 1 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 2 \end{pmatrix}.$$

It can be seen that the homogeneous congruence system

$$\hat{\mathcal{C}} = \{2x - 4x_0 \equiv_8 0, -x + 2y + 4x_0 \equiv_8 0, x - y + 2z - 7x_0 \equiv_8 0, 8x_0 \equiv_8 0\}$$

that corresponds to the matrix $\hat{F}$ is not in strong minimal form. This can be seen if we take the congruence $2x - 4x_0 \equiv_8 0$ which has pivot variable x and coefficient 2. Then for $\hat{\mathcal{C}}$ to be in strong minimal form, for all other congruences, the coefficient for the x variable should be greater than -1 and less than or equal to 1, however this is not the case for the congruence $-x + 2y + 4x_0 \equiv_8 0$.

Note that, we could also consider the congruence $-x + 2y + 4x_0 \equiv_8 0$, which has pivot variable y and coefficient 2. Then for $\hat{\mathcal{C}}$ to be in strong minimal form, for all other congruences the coefficient for the y variable should be greater than -1 and less than or equal to 1, however this is not the case for the congruence $x-y+2z-7x_0 \equiv_8 0$. Also as the congruence $8x_0 \equiv_8 0$ has pivot variable x_0 and coefficient 8, for all other congruences the coefficient for the x_0 variable should be greater than -4 and less than or equal to 4, however this is not the case for the congruences $2x - 4x_0 \equiv_8 0$ and $x - y + 2z - 7x_0 \equiv_8 0$.

3.6 Double Description

We have shown that any grid $\mathcal{L}$ can be described by using a congruence system $\mathcal{C}$ or generated by a generator system $\mathcal{G}$. Therefore, just as for the double description method for convex polyhedra, since we have shown we have the algorithms for converting a representation of one kind into a representation of the other kind and for minimising both representations, we can represent the grid $\mathcal{L}$ by the *double description* $(\mathcal{C}, \mathcal{G})$. Note that, if $(\mathcal{C}, \mathcal{G})$ is a double description for a grid and $\hat{\mathcal{C}}$ and $\hat{\mathcal{G}}$ are homogeneous forms for $\mathcal{C}$ and $\mathcal{G}$, then $(\hat{\mathcal{C}}, \hat{\mathcal{G}})$ is also a double description.

Suppose we have a double description $\big(\mathcal{C}, \mathcal{G}\big)$ of a grid $\mathcal{L} \in \mathbb{G}_n$, where both $\mathcal{C}$ and $\mathcal{G}$ are in minimal form. Then, it follows from the definition of minimal form that $\#\,\mathcal{C} \leq n + 1$ and $\#\,L + \#\,Q \leq n$. In fact, we have a stronger result.

Proposition 3.32 *Let $(\mathcal{C}, \mathcal{G})$ be a double description where both $\mathcal{C}$ and $\mathcal{G}$ are in minimal form. Letting $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$, where $\mathcal{E}$ and $\mathcal{F}$ are sets of equalities and proper congruences, respectively, and $\mathcal{G} = (L, Q, P)$, then $\#\,\mathcal{F} = \#\,Q = n - \#\,L - \#\,\mathcal{E}$.*

Example 3.33 *Consider the grid $\mathcal{L}$ from Example 3.2 and Example 3.11 which can be seen in Figure 3.1 and Figure 3.3. The congruence system $\mathcal{C}$ and the generator system $\mathcal{G}_2$ are in minimal*

form; however, $\mathcal{G}_1$ is not as it contains more than one point. Furthermore, for $i = 1, 2$, the pairs $(\mathcal{C}, \mathcal{G}_i)$ are double descriptions for $\mathcal{L}$.

The proof of Proposition 3.32 depends on the following lemma. This shows that if one grid is a subset of another then the pivot elements of the proper congruences of the larger grid must be divisible by the corresponding pivot elements of the smaller grid.

Lemma 3.34 *Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ be non-empty grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are in minimal form. Then, for each $\gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_g d\big) \in \mathcal{C}_2$, there exists $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}_1$ such that $\mathrm{piv}_<(\mathbf{a}) = \mathrm{piv}_<(\mathbf{c}) = k$ and either $f = g = 0$ or $g \neq 0$ and $a_k \mid f c_k$.*

Proof. Suppose $\gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_g d\big) \in \mathcal{C}_2$ and $\mathrm{piv}_<(\mathbf{c}) = k$. Then, as $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$. Let $\mathcal{G}_1 = \big(L_1, Q_1, \{\mathbf{p}\}\big)$ be a generator system for $\mathcal{L}_1$ in minimal form constructed as in Lemma 3.28 from $\mathcal{C}_1$.

We first prove that there exists $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}_1$ such that $\mathrm{piv}_<(\mathbf{a}) = k$. To see this, suppose instead that, for all $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}_1$, $\mathrm{piv}_<(\mathbf{a}) \neq k$. Then, by Lemma 3.28, there must exist a line $\boldsymbol{\ell} \in L_1$ such that $\mathrm{piv}_>(\boldsymbol{\ell}) = k$; hence $\langle \mathbf{c}, \boldsymbol{\ell}\rangle = c_k \ell_k \neq 0$. Since $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$, this implies that

$$\big\langle \mathbf{c}, (\mathbf{p} + r\boldsymbol{\ell})\big\rangle = \langle \mathbf{c}, \mathbf{p}\rangle + r c_k \ell_k \equiv_g d,$$

for all $r \in \mathbb{R}$, which is a contradiction.

We next show that if $g = 0$ then $f = 0$. To see this, suppose instead that $g = 0$ but $f \neq 0$. Then, by Lemma 3.28, there exists $\mathbf{q} \in Q_1$ such that $\mathrm{piv}_>(\mathbf{q}) = k$; hence $\langle \mathbf{c}, \mathbf{q}\rangle = c_k q_k \neq 0$. Since $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$, this implies that

$$\big\langle \mathbf{c}, (\mathbf{p} + m\mathbf{q})\big\rangle = \langle \mathbf{c}, \mathbf{p}\rangle + m c_k q_k \equiv_g d,$$

for all $m \in \mathbb{Z}$, which is a contradiction.

We now assume that $g \neq 0$ and show that $a_k \mid f c_k$. This is trivial if $f = 0$; therefore, suppose $f = 1$. By Lemma 3.28, there exists a parameter $\mathbf{q}$ in Q_1 such that $\mathrm{piv}_>(\mathbf{q}) = k$ (so that $q_k c_k \neq 0$) and $q_k = a_k^{-1}$. Thus, as $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$, $\langle \mathbf{q}, \mathbf{c}\rangle = q_k c_k = m$, for some $m \in \mathbb{Z} \setminus \{0\}$. Therefore we must have $a_k \mid f c_k$. $\square$

Proof [of Proposition 3.32] Let $\mathcal{C}'$ be the congruence system obtained, as in Lemma 3.28, from $\mathcal{G}$. Let $\mathcal{G} = (L, Q, P)$ and let $\mathcal{C} = (\mathcal{F}, \mathcal{E})$ and $\mathcal{C}' = (\mathcal{F}', \mathcal{E}')$ where $\mathcal{E}$ and $\mathcal{E}'$ are sets of equalities, and $\mathcal{F}$ and $\mathcal{F}'$ are sets of proper congruences. Then, by Lemma 3.28,

$$\#\, Q = \#\, \mathcal{F}' = n - \#\, L - \#\, \mathcal{E}'.$$

By applying Lemma 3.34 twice with $\mathcal{L}_1 = \mathcal{L}_2 = \mathcal{L}$, we obtain $\#\mathcal{E} = \#\mathcal{E}'$ and $\#\mathcal{F} = \#\mathcal{F}'$. Therefore

$$\#Q = \#\mathcal{F} = n - \#L - \#\mathcal{E}.$$

□

3.7 Implementation

The domain of grids is fully supported and implemented within the Parma Polyhedra Library (PPL) [11, 13]. The PPL is a C++ library which can manipulate numerical information that can be represented by vectors of an n-dimensional space. As well as the grid domain the PPL also supports the domain of convex polyhedra and bounded difference shapes. Among the tests available in the PPL are the examples in [3] and implementations of the running examples in [60, 61]. The PPL provides full support for lifting any domain to the powerset of that domain, so that a user of the PPL can experiment with powersets of grids and the extra precision this provides.

3.8 Related Work

In [37], Granger introduces a simple integer *non-relational* grid domain, which he calls a congruence analysis, that is a grid described by congruences of the form $x = b \pmod f$ where b and f are integers. He shows in examples how a static analysis can infer congruence information and show that this domain can obtain more precise information for applications such as automatic vectorization. In the Master Thesis of Bygde [19] it is shown that the domain of integer rectilinear grids, based on that introduced in by Granger [37] and extended to include bit-level operators, can be used to estimate the worst case execution time (WCET) of a program given a specific system. Larsen et al. [47] have also developed a static analyzer over a non-relational grid domain specifically designed to detect when dynamic memory addresses are congruent with respect to a given modulus; they show that this information helps in the construction of a comprehensive set of program transformations for saving energy on low-power architectures and improving performance on multimedia processors. We note that these applications should carry over to the more complex domain considered here. In addition, Miné has shown how to construct, from the non-relational congruence domain in [37], a zone-congruence domain. This domain only allows *weakly relational* congruences, that is congruences that have the form $x - y = b \pmod f$ where b and f are rationals [51, 53]. The set of congruences is then represented by a constraint matrix like that for a bounded difference shape or octagon and the operations are then applied to this matrix. In [51] Miné gives an algorithm for producing the closure of a system of congruence constraints, based on the *Floyd-Warshall algorithm* [24], with complexity $\mathrm{O}(n^2)$ if one congruence constraint is added. If however all n^2 possible congruence constraints are added the complexity becomes $\mathrm{O}(n^4)$.

With regard to the *fully relational* domains, note that the use of a domain of linear *equality*

relations for program analysis had been studied by Karr [45]. In [39], Granger generalized this to provide a domain of linear *congruence* relations on an integral domain, i.e., a domain generated by integral vectors in n-dimensions only instead of rationals; and then, in [38, 41], Granger generalized the results to the full grid domain over the rationals. In [38, 39, 41], domain elements are represented by both congruence and generator systems similar to the ones defined here. Standard algorithms for solving linear equations are used in converting from generator to congruence systems; however, a more complex $\mathrm{O}(n^4)$ algorithm is provided for converting from congruence to generator systems. This is because the congruences are converted and added one at a time to the new minimised generator system. Assuming the number of generators is $n + 1$, the algorithm for minimising the generator system has complexity $\mathrm{O}(n^3 \log_2 n)$.

The problem of how the grid domain can be applied in a program analyzer has been studied by Müller-Olm and Seidl in [58,59,61] also building on the work of Karr [45]. Here, the prime focus is for the design of an *interprocedural* analysis for programs containing assignment statements and procedure calls. The algorithm has three stages: first, for each program point, a matrix M containing a (minimised) set of generators (i.e., vectors of values that hold at that point) is found; secondly, the determinant f of M is computed; thirdly, a congruence system with modulo f that satisfies all the vectors in M is determined. Stage one is similar to that proposed by Granger [39] for minimising a set of generators. Stages two and three differ from the conversion in [39] in that the modulus f is computed separately and used to reduce the sizes of the coordinates. Also in [60, 62] they consider the specific case of congruence equations where the modulo is a power of 2. Again this work is mainly performed over the set of generators and all algorithms have the same complexities as those mentioned in [58, 59, 61]. It is noted that this paper overlooks the work of Granger on rational congruence equations [38, 41]. Note that the framework described in [62] subsumes previous works by the same authors. From this work on congruence equations with modulo a power of two, King and Sondergaard [46] consider using a SAT solver to derive these equations which contain information about non-linear operations within the program.

Following an independent stream of research, Ancourt [1] in her thesis considered the domain of $\mathbb{Z}$*-polyhedra*; that is a domain of *integral lattices* intersected with the domain of convex polyhedra (see also [68, 71, 72]). As we are primarily interested here in the "integral lattices" component which may be seen as a sub-domain of the domain of grids where the grid is full dimensional, does not contain lines in the representation and all the grid points are integral vectors. The representation of these integral lattices is a special case of our generator representation where, for n dimensions, there must be exactly one point and n linearly independent parameters, all of which must be integral. There is no support so far for a congruence representation. All the operations on $\mathbb{Z}$-polyhedra (and therefore the lattices) require canonical representations; hence Quinton et al. [71, 72] define a canonical form for these lattices with a method for its computation. We note that the algorithm for computing the canonical form has complexity $\mathrm{O}(n^4)$, where n is the number of dimensions of the vector space.

As shown in Section 3.5 the *homogeneous form* of a representation is required by the conver-

sion algorithm. This homogeneous form is not new, in fact several researchers have observed this. For instance, Granger [39] describes a map from a linear congruence system in n variables to a homogeneous one in $n+1$ variables; Nookala and Risset [68] explain that the PolyLib [48] adds a dimension to make the generator representation homogeneous; while Müller-Olm and Seidl [61] consider *extended states* where vectors have an extra 0'th component.

The Hermite Normal Form algorithm [67, 76] for lattices is sufficient to ensure a representation is in strong minimal form, however as we wish to ensure that the coefficients are as small as possible in absolute value we use a different requirement. That is, if $\mathcal{C}$ is in minimal form and, for each pair of distinct proper congruences

$$\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_1 b\big),\ \gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_1 d\big) \in \mathcal{C},$$

if $\mathrm{piv}_<(\gamma) = k > 0$, then $-c_k < 2a_k \leq c_k$. Where as for the Hermite Normal Form the coefficient a_k is bounded by $0 \leq a_k < c_k$. Similarly for the coefficients of the generator system.

3.9 Conclusion

In this chapter we have presented the domain of Grids. We have shown that the domain may be represented by either a set of congruences or a set of generators. We introduced 2 methods for minimising the representation of a grid, the minimal form which has complexity $\mathrm{O}\big(n^2 m\big)$, which is better than previous proposals [39, 61, 62], and the strong minimal form which has complexity $\mathrm{O}\big(n^3\big)$. We have shown how conversion can be implemented using any matrix inversion algorithm, inheriting the corresponding worst-case complexity. For instance, the complexity is $\mathrm{O}\big(n^3\big)$ when adopting the standard Gaussian elimination method. Previous proposals for congruence to generator conversion have complexity no better than $\mathrm{O}\big(n^4\big)$ [41].

Chapter 4

The Grid Domain Operations

4.1 Introduction

In this chapter we introduce the main operations for the domain of grids. These abstract operations will be based on some of the set-theoretic operations, such as containment, intersection, union and difference. We will show that by taking set-theoretic operations we do not always produce a single grid and therefore the abstract operations compute an approximation of them.

4.2 Comparison

In this section we show how to test if two grids are equal or if one grid is contained in another. This is important since we need to be able to check if a fixpoint has been reached or to model an inequality or equality test in a program.

For any pair of grids $\mathcal{L}_1 = \mathrm{ggen}(L, Q, P)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C})$ in $\mathbb{G}_n$, we can decide whether $\mathcal{L}_1 \subseteq \mathcal{L}_2$ by checking if every generator in (L, Q, P) satisfies every congruence in $\mathcal{C}$. Note that a point $\mathbf{p}$ *satisfies* a congruence $\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b$ if $\langle \mathbf{a}, \mathbf{p} \rangle \equiv_f b$ and a parameter or line $\mathbf{v}$ *satisfies* a congruence $\langle \mathbf{a}, \mathbf{x} \rangle \equiv_f b$ if $\langle \mathbf{a}, \mathbf{v} \rangle \equiv_f 0$. Let $\mathcal{G} = (L, Q, P)$, $m_1 = \#L + \#Q + \#P$ and $m_2 = \#\mathcal{C}$. Then assuming that the systems $\mathcal{G}$ and $\mathcal{C}$ are already available, each of the m_1 generators must be checked against the m_2 congruences. Hence there are $m_1 m_2$ checks to be made and each check requires $\mathrm{O}(n)$ arithmetical operations. Therefore the worst-case complexity of comparing two grids is $\mathrm{O}(m_1 m_2 n)$. Note that, if $n \leq \min\{m_1, m_2\}$, then it would be computationally more efficient to compute the minimal forms for $\mathcal{C}$ and $\mathcal{G}$ before actually checking for comparison. This is because the complexity of the minimisations would be $\mathrm{O}(m_1 n^2)$ for the generator system and $\mathrm{O}(m_2 n^2)$ for the congruence system, which are less than or equal to $\mathrm{O}(m_1 m_2 n)$

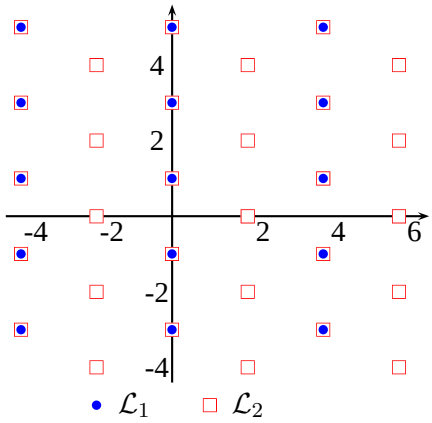


Figure 4.1: Comparing two grids in $\mathbb{R}^2$.

for $n \leq \min\{m_1, m_2\}$. Hence obtaining the worst-case complexity $\mathrm{O}\big(n^2 \max\{m_1, m_2\}\big)$ if $n \leq \min\{m_1, m_2\}$. Finally if the generator and congruence systems are already available in minimal form the complexity of comparison is $\mathrm{O}\big(n^3\big)$.

Example 4.1 *Let* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1) = \mathrm{ggen}(\mathcal{G}_1)$ *and* $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{ggen}(\mathcal{G}_2)$ *in* $\mathbb{G}_2$ *where*

$$\mathcal{C}_1 := \{x \equiv_4 0,\ y \equiv_2 1\} \quad \textit{and} \quad \mathcal{C}_2 := \{x \equiv_2 0, -x + 2y \equiv_4 2\}.$$

$$\mathcal{G}_1 := \left(\varnothing, \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) \quad \textit{and} \quad \mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right).$$

Then it can be seen in Figure 4.1 that $\mathcal{L}_1 \subseteq \mathcal{L}_2$ *where grid* $\mathcal{L}_1$ *is illustrated in Figure 4.1 by the filled circles and the grid* $\mathcal{L}_2$ *is illustrated in Figure 4.1 by the square points.*

If it is known that one grid is a subset of another, then, assuming that the descriptions of both grids are available in minimal form, there are more efficient tests for checking equality which are shown in the following Propositions.

Proposition 4.2 *Let* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ *and* $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ *be non-empty grids in* $\mathbb{G}_n$*, where the congruence systems* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are in minimal form. Suppose also that* $\mathcal{L}_1 \subseteq \mathcal{L}_2$*, then* $\mathcal{L}_1 = \mathcal{L}_2$ *if and only if* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are pivot equivalent.*

Proposition 4.3 *Let* $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ *and* $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$ *be non-empty grids in* $\mathbb{G}_n$*, where the generator systems* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are in minimal form. Suppose also that* $\mathcal{L}_1 \subseteq \mathcal{L}_2$*, then* $\mathcal{L}_1 = \mathcal{L}_2$ *if and only if* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are pivot equivalent.*

We require the condition $\mathcal{L}_1 \subseteq \mathcal{L}_2$ in Propositions 4.2 and 4.3 since suppose we have two grids $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1) = \mathrm{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{ggen}(\mathcal{G}_2)$. Then if it is known that $\mathcal{C}_1 \Uparrow \mathcal{C}_2$ or $\mathcal{G}_1 \Downarrow \mathcal{G}_2$ then we cannot deduce that $\mathcal{L}_1 = \mathcal{L}_2$ unless we know that $\mathcal{L}_1 \subseteq \mathcal{L}_2$. The following example illustrates this.

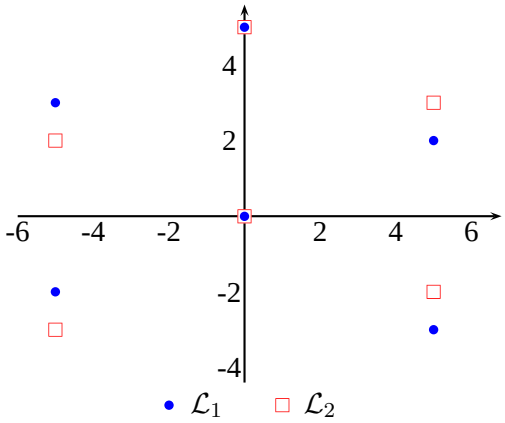


Figure 4.2: The equality test with a missing condition.

Example 4.4 *Consider the grids* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1) = \mathrm{ggen}(\mathcal{G}_1)$ *and* $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{ggen}(\mathcal{G}_2)$ *in* $\mathbb{G}_n$*, where*

$$\mathcal{C}_1 = \{5x \equiv_{25} 0, -2x + 5y \equiv_{25} 0\}, \qquad \mathcal{C}_2 = \{5x \equiv_{25} 0, -3x + 5y \equiv_{25} 0\}$$
$$\mathcal{G}_1 = \left(\varnothing, \begin{pmatrix} 5 & 0 \\ 2 & 5 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right), \qquad \mathcal{G}_2 = \left(\varnothing, \begin{pmatrix} 5 & 0 \\ 3 & 5 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right).$$

Then $\mathcal{C}_1 \Uparrow \mathcal{C}_2$ *and* $\mathcal{G}_1 \Downarrow \mathcal{G}_2$*. It can be seen in Figure 4.2 however that* $\mathcal{L}_1 \neq \mathcal{L}_2$*.*

The next lemma, needed for the proof of Proposition 4.2 and Proposition 4.3, shows that if, two grids, one a subset of the other are described by two congruence systems in strong minimal form that are pivot equivalent, then, relative to the affine hull of the grids, pivot equivalent congruences in these systems are the same.

Lemma 4.5 *Let* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$*,* $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ *be non-empty grids in* $\mathbb{G}_n$ *where* $\mathcal{L}_1 \subseteq \mathcal{L}_2$ *and the congruence systems* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are in strong minimal form. Suppose that* $\mathcal{C}_1$ *is pivot equivalent to* $\mathcal{C}_2$*. Then, for each* $\beta \in \mathcal{C}_1$ *and* $\gamma \in \mathcal{C}_2$ *such that* $\beta \Uparrow \gamma$*,*

$$\mathrm{gcon}\big(\{\beta\}\big) \cap \mathrm{affine.hull}(\mathcal{L}_1) = \mathrm{gcon}\big(\{\gamma\}\big) \cap \mathrm{affine.hull}(\mathcal{L}_1). \tag{4.1}$$

Proof. Let $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}_1$. By the definition of pivot equivalence for congruence systems in Section 3.5, as $\mathcal{C}_1 \Uparrow \mathcal{C}_2$ there exists $\gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_g d\big) \in \mathcal{C}_2$ such that $\beta \Uparrow \gamma$. We show that equation (4.1) holds. By the definition of pivot equivalence for congruences in Section 2.1, $\mathrm{piv}_<(\mathbf{a}) = \mathrm{piv}_<(\mathbf{c}) = k$ and $ga_k = fc_k$. Thus as $a_k, c_k \neq 0$, either $f = g = 0$ and β, γ are both equalities, or we have $f, g \neq 0$ so that β, γ are both proper congruences and we can assume that $f = g = 1$.

Let $\mathcal{E}_1$ be the set of equalities in $\mathcal{C}_1$. By Gaussian elimination, the set $\mathcal{E}_1$ can be transformed to the set of equalities $\mathcal{E}_1'$ such that $\mathrm{gcon}(\mathcal{E}_1') = \mathrm{gcon}(\mathcal{E}_1) = \mathrm{affine.hull}(\mathcal{L}_1)$ and has the following

property: let $\mathcal{E}'_1 = \{\beta_1, \ldots, \beta_m\}$ such that, for each $i \in \{1, \ldots, m\}$, $\mathrm{piv}_<(\beta_i) = k_i$ and $\beta_i = \big(\langle \mathbf{a}_i, \mathbf{x}\rangle = b_i\big)$; then, for each $i, j \in \{1, \ldots, m\}$ where $i \neq j$, we have $a_{ik_j} = 0$. Let

$$\mathbf{a}'' = \mathbf{a} - \mathbf{c} - \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} \mathbf{a}_i, \qquad b'' = b - d - \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} b_i. \tag{4.2}$$

Let $\beta'' := \big(\langle \mathbf{a}'', \mathbf{x}\rangle \equiv_1 b''\big)$ and $\ell := \mathrm{piv}_<(\mathbf{a}'')$. Then $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\beta''\}\big) \subseteq \mathcal{L}_2$. Moreover, for any equality $\beta_i \in \mathcal{E}'_1$, $\mathrm{piv}_<(\mathbf{a}_i) \neq \ell$. Thus, if β, γ are equalities, $\mathbf{a}'' = \mathbf{0}$ and, as $\mathcal{C}_1$ is consistent, $b'' = 0$. Therefore $\mathrm{gcon}(\mathcal{E}'_1) \subseteq \mathrm{gcon}\big(\{\beta\}\big)$ and $\mathrm{gcon}(\mathcal{E}'_1) \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$. Hence equation (4.1) holds.

Consider now the case when β, γ are proper congruences. We first show that $\mathbf{a}'' = \mathbf{0}$ and $b'' \in \mathbb{Z}$. Without loss of generality we can assume that $f = g = 1$. Note that, as $a_k = c_k$ we have $\ell < k$. We show $\ell = 0$; suppose, to the contrary that $\ell > 0$. Since $\mathcal{L}_1 \subseteq \mathcal{L}_2$, and $\gamma \in \mathcal{C}_2$, we have $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$; so we can apply Lemma 3.34 to the grids $\mathcal{L}_1$ and $\mathrm{gcon}\big(\{\gamma\}\big)$. Thus there exists a proper congruence $\beta' = \big(\langle \mathbf{a}', \mathbf{x}\rangle \equiv_1 b'\big) \in \mathcal{C}_1$ where $\mathrm{piv}_<(\mathbf{a}') = \ell$ and $a'_\ell \mid a''_\ell$. Note that the number of proper congruences p_1 in $\mathcal{C}_1$ is equal to the number of proper congruences p_2 in $\mathcal{C}_2$; since by Lemma 3.34, $p_2 \leq p_1$ and, by hypothesis, $p_1 \leq p_2$. Therefore, by Lemma 3.34, there must exist a proper congruence $\gamma' = \big(\langle \mathbf{c}', \mathbf{x}\rangle \equiv_1 d'\big) \in \mathcal{C}_2$ where $\mathrm{piv}_<(\mathbf{c}') = \ell$ and $c'_\ell = a'_\ell$. Now as $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form, by Definition 3.21,

$$-\frac{a'_\ell}{2} < a_\ell \leq \frac{a'_\ell}{2} \quad \text{and} \quad -\frac{c'_\ell}{2} < c_\ell \leq \frac{c'_\ell}{2}.$$

Therefore $-a'_\ell < a''_\ell < a'_\ell$. It follows that, as $a'_\ell | a''_\ell$, $a''_\ell = 0$, contradicting the assumption that $\mathrm{piv}_<(\mathbf{a}'') = \ell > 0$. Therefore $\mathbf{a}'' = \mathbf{0}$ and β'' is the relation $b'' \equiv_1 0$ for some $b'' \in \mathbb{Z}$.

It follows that, by (4.2),

$$\mathbf{a} - \mathbf{c} = \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} \mathbf{a}_i, \qquad b - d \equiv_1 \sum_{i=1}^{m} \frac{(a_{k_i} - c_{k_i})}{a_{k_i}} b_i.$$

Thus

$$\mathrm{gcon}\big(\{\gamma, \beta_1, \ldots, \beta_m\}\big) \subseteq \mathrm{gcon}\big(\{\beta\}\big) \quad \text{and} \quad \mathrm{gcon}\big(\{\beta, \beta_1, \ldots, \beta_m\}\big) \subseteq \mathrm{gcon}\big(\{\gamma\}\big)$$

so that

$$\mathrm{gcon}\big(\{\gamma, \beta_1, \ldots, \beta_m\}\big) = \mathrm{gcon}\big(\{\beta, \beta_1, \ldots, \beta_m\}\big).$$

Hence equation (4.1) holds. □

From this result we can now prove Propositions 4.2 and 4.3.

Proof [of Proposition 4.2] First we show that if the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ are in minimal

form and pivot equivalent, then $\mathcal{L}_1 = \mathcal{L}_2$. By Proposition 3.22, we can convert $\mathcal{C}_1$ and $\mathcal{C}_2$ to strong minimal form, $\mathcal{C}_1'$ and $\mathcal{C}_2'$ respectively, so that, for $i = 1, 2$, $\mathcal{L}_i = \mathrm{gcon}(\mathcal{C}_i')$ and $\mathcal{C}_i'$ is pivot equivalent to $\mathcal{C}_i$. By hypothesis, $\mathcal{L}_1 \subseteq \mathcal{L}_2$. Thus, by Lemma 4.5, for each $\beta \in \mathcal{C}_1'$ and $\gamma \in \mathcal{C}_2'$,

$$\mathrm{gcon}\big(\{\beta\}\big) \cap \mathrm{affine.hull}(\mathcal{L}_1) = \mathrm{gcon}\big(\{\gamma\}\big) \cap \mathrm{affine.hull}(\mathcal{L}_1).$$

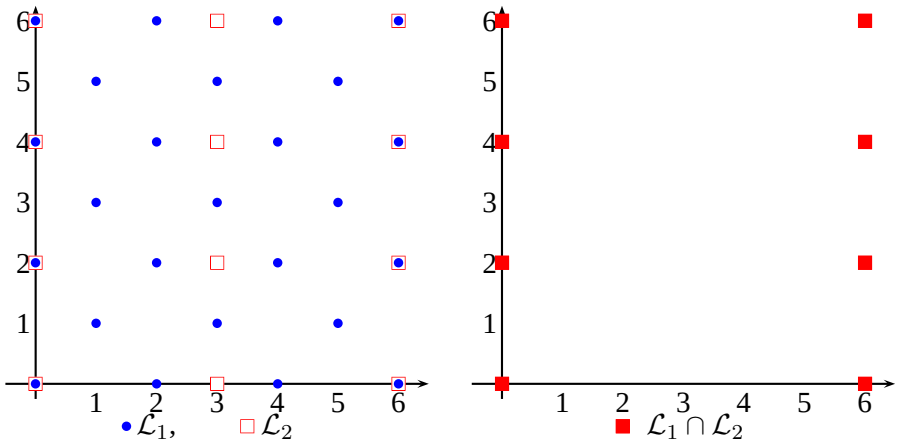
Thus $\mathcal{L}_1 = \mathcal{L}_2$, as required.

We now assume that $\mathcal{L}_1 = \mathcal{L}_2$. Suppose that the congruence systems $\mathcal{C}_1, \mathcal{C}_2$ are in minimal form; then we show that $\mathcal{C}_1$ and $\mathcal{C}_2$ are pivot equivalent. Let $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}_1$. Then as $\mathcal{L}_2 \subseteq \mathcal{L}_1$, by Lemma 3.34, there exists $\gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_g d\big) \in \mathcal{C}_2$ such that $\mathrm{piv}_<(\mathbf{a}) = \mathrm{piv}_<(\mathbf{c}) = k$ and either $f = g = 0$ or $f \neq 0$ and $c_k \mid g a_k$. Also, as $\mathcal{L}_1 \subseteq \mathcal{L}_2$, by Lemma 3.34, and property (2) of Definition 3.15, if $g \neq 0$, then $a_k \mid f c_k$. Therefore if $f \neq 0$ and $g \neq 0$ we can assume without loss of generality that $f = g = 1$. By property (1) of Definition 3.15, $a_k, c_k > 0$ so that we have $a_k = c_k$. Hence $\mathcal{C}_1$ and $\mathcal{C}_2$ are pivot equivalent. □

Proof [of Proposition 4.3] First we show that if the generator systems $\mathcal{G}_1$ and $\mathcal{G}_2$ are in minimal form and pivot equivalent, then $\mathcal{L}_1 = \mathcal{L}_2$. Let $\mathcal{C}_1'' = (\mathcal{F}_1'', \mathcal{E}_1'')$ and $\mathcal{C}_2'' = (\mathcal{F}_2'', \mathcal{E}_2'')$ be congruence systems for $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, as constructed in Lemma 3.28 from the generator systems $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. Then by properties (7) and (8) in Lemma 3.28, $\mathcal{C}_1''$ is pivot equivalent to $\mathcal{C}_2''$. Thus, by Proposition 4.2, $\mathcal{L}_1 = \mathcal{L}_2$, as required.

Finally, suppose that the generator systems $\mathcal{G}_1 = (L_1, Q_1, P_1)$ and $\mathcal{G}_2 = (L_2, Q_2, P_2)$ are in minimal form; then we show that $\mathcal{G}_1$ and $\mathcal{G}_2$ are pivot equivalent. Let $\mathcal{C}_1'' = (\mathcal{F}_1'', \mathcal{E}_1'')$ and $\mathcal{C}_2'' = (\mathcal{F}_2'', \mathcal{E}_2'')$ be congruence systems as constructed in Lemma 3.28 from the generator systems $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. Then $\mathcal{C}_1'', \mathcal{C}_2''$ are in minimal form and, by Proposition 4.2, $\mathcal{C}_1''$ and $\mathcal{C}_2''$ are pivot equivalent. Suppose $\mathbf{v} \in Q_1 \cup P_1$ and that $\mathrm{piv}_>(\mathbf{v}) = k$. Then, by property (7) of Lemma 3.28, there exists $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_1 0\big) \in \mathcal{F}_1''$ such that $\mathrm{piv}_<(\mathbf{a}) = k$ and $v_k a_k = 1$. By Definition 3.23, there exists $\gamma = \big(\langle \mathbf{c}, \mathbf{x}\rangle \equiv_1 0\big) \in \mathcal{F}_2''$ such that $\mathrm{piv}_<(\mathbf{c}) = k$ and $a_k = c_k$. By property (7) of Lemma 3.28, there exists $\mathbf{w} \in Q_2 \cup P_2$ such that $w_k c_k = 1$. Hence $v_k = w_k$. Suppose next $\mathbf{v} \in L_1$ and that $\mathrm{piv}_>(\mathbf{v}) = k$. By Proposition 3.32, $\#\,\mathcal{C}_1'' = n - \#\,L_1$ so that, by Definition 3.18, for all $\beta \in \mathcal{C}_1''$, we have $\mathrm{piv}_<(\beta) \neq k$. By Definition 3.23, for all $\gamma \in \mathcal{C}_2''$, $\mathrm{piv}_<(\gamma) \neq k$. Also by Proposition 3.32, $\#\,\mathcal{C}_2'' = n - \#\,L_2$ so that, by Definition 3.18, there exists $\mathbf{w} \in L_2$ such that $\mathrm{piv}_>(\mathbf{w}) = k$. Hence $\mathcal{G}_1$ and $\mathcal{G}_2$ are pivot equivalent. □

It follows from Proposition 4.2 and Proposition 4.3, that provided $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and $\mathcal{L}_1$ and $\mathcal{L}_2$ have both their generator or congruence systems already in minimal form, then the complexity of checking if $\mathcal{L}_1 = \mathcal{L}_2$ is just $\mathrm{O}\big(n\big)$. Note that, the computational cost is low due to the fact that, for this quick check, each elementary operation is a comparison between two numbers. It also follows from Proposition 4.2 and Proposition 4.3, that, if it is found that one pair of corresponding pivot elements of the congruence or generator systems differ, then we can immediately deduce that the grids they describe also differ.



(a) Grids $\mathcal{L}_1$ and $\mathcal{L}_2$. (b) Grid $\mathcal{L}_1 \cap \mathcal{L}_2$.

Figure 4.3: Grid intersection.

4.3 Intersection

We will now introduce the operation of intersection, we require this operation in, for example, data dependence analysis for arrays [70]. For two grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *intersection* of $\mathcal{L}_1$ and $\mathcal{L}_2$ is defined as the set intersection $\mathcal{L}_1 \cap \mathcal{L}_2$, which can also be thought of as the largest grid included in both $\mathcal{L}_1$ and $\mathcal{L}_2$. In theoretical terms, the intersection operation is the binary *meet* operator on the lattice $\mathbb{G}_n$. If $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2)$, then the intersection can be computed by $\mathcal{L}_1 \cap \mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_1 \cup \mathcal{C}_2)$.

The cost of computing the grid intersection depends on a number of factors. If the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, are known, then the complexity of computing $\mathcal{L}_1 \cap \mathcal{L}_2$ is linear in either $\#\,\mathcal{C}_1$ or $\#\,\mathcal{C}_2$ as the congruences of one system are mapped to the other system of congruences. If, however, only the generator systems $\mathcal{G}_1$ and $\mathcal{G}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, are known and are not necessarily in minimal form, then the complexity of intersection is that of minimising and converting the generator systems which is, at worst, $\mathrm{O}\big(n^2 m\big)$, where $m = \max(\#\,\mathcal{G}_1, \#\,\mathcal{G}_2, n)$. A computation of grid intersection is given in Example 4.6.

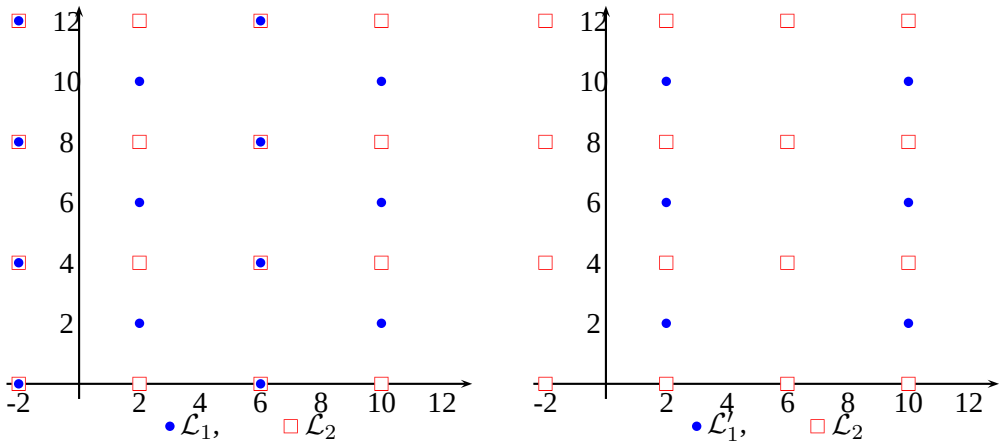
Example 4.6 *Consider the grids $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2)$ in $\mathbb{G}_2$ where*

$$\mathcal{C}_1 := \{x \equiv_1 0,\ x + y \equiv_2 0\} \quad \textit{and} \quad \mathcal{C}_2 := \{x \equiv_3 0,\ y \equiv_2 0\}.$$

The grids $\mathcal{L}_1$ and $\mathcal{L}_2$ are illustrated by the filled circles and open squares, respectively, in Figure 4.3(a). Then the grid intersection is $\mathcal{L}_1 \cap \mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_1 \cup \mathcal{C}_2)$. The minimal form of the congruence system $\operatorname{gcon}(\mathcal{C}_1 \cup \mathcal{C}_2)$ is $\mathcal{C} = \{x \equiv_6 0,\ y \equiv_2 0\}$, thus $\mathcal{C}$ is a minimal form of $\mathcal{L}_1 \cap \mathcal{L}_2$. Therefore, we have

$$\mathcal{L}_1 \cap \mathcal{L}_2 = \{x \equiv_6 0,\ y \equiv_2 0\}.$$

The grid $\mathcal{L}_1 \cap \mathcal{L}_2$ is illustrated by the filled squares in Figure 4.3(b).



(a) Grids $\mathcal{L}_1$ and $\mathcal{L}_2$. (b) The set $\mathcal{L}_1 \cup \mathcal{L}_2$.

Figure 4.4: The union of two grids.

4.4 Join

We will now introduce the operation of join which will approximate a set-theoretic union, we require this operation if, for example, we had a program fragment that split into two separate threads. The reason we do not actually take a set-theoretic union is that the result would not always be a single grid but often represented by a disjoint union of grids. Example 4.7 shows this.

Example 4.7 *Consider* $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ *and* $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$ *in* $\mathbb{G}_2$*, where*

$$\mathcal{G}_1 := \left(\varnothing, \begin{pmatrix} 4 & 0 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix}\right) \quad \textit{and} \quad \mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right).$$

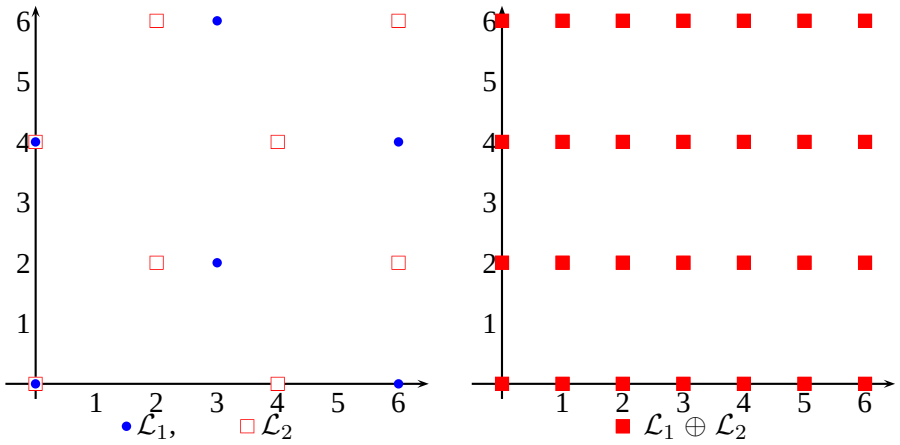
The grids $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *are illustrated by the filled circles and open squares, respectively, in Figure 4.4(a). Then there is no single grid that can represent* $\mathcal{L}_1 \cup \mathcal{L}_2$ *exactly. Instead* $\mathcal{L}_1 \cup \mathcal{L}_2$ *can be represented by the union of two disjoint grids, namely* $\mathcal{L}'_1 = \mathrm{ggen}(\mathcal{G}'_1)$ *and* $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$*, where*

$$\mathcal{G}'_1 := \left(\varnothing, \begin{pmatrix} 8 & 0 \\ 0 & 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix}\right).$$

The grids $\mathcal{L}'_1$ *and* $\mathcal{L}_2$ *are illustrated by the filled circles and open squares, respectively, in Figure 4.4(b).*

For grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *grid join* of $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \oplus \mathcal{L}_2$, is the smallest grid that includes both $\mathcal{L}_1$ and $\mathcal{L}_2$. The grid join operator is the binary *join* operator on the lattice $\mathbb{G}_n$. If $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$, then the grid join is computed by $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_1 \cup \mathcal{G}_2)$.

The cost of computing the grid join depends on a number of factors. If the generator systems $\mathcal{G}_1$ and $\mathcal{G}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, are known, then the complexity of computing $\mathcal{L}_1 \oplus \mathcal{L}_2$ is linear in either $\#\mathcal{G}_1$ or $\#\mathcal{G}_2$ as one set of generators is mapped to the other generator



(a) Grids $\mathcal{L}_1$ and $\mathcal{L}_2$. (b) Grid $\mathcal{L}_1 \oplus \mathcal{L}_2$.

Figure 4.5: Grid join.

system. If, however, only the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, are known and are not necessarily in minimal form, then the complexity is that of minimising and converting the congruence systems which is, at worst, $\mathrm{O}(n^2 m)$, where $m = \max(\#\,\mathcal{C}_1, \#\,\mathcal{C}_2, n)$. A computation of grid join is given in Example 4.8.

Example 4.8 *Consider* $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ *and* $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$ *in* $\mathbb{G}_2$*, where*

$$\mathcal{G}_1 := \left(\varnothing, \begin{pmatrix} 3 & 0 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right) \quad \textit{and} \quad \mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right).$$

The grids $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *are illustrated by the filled circles and open squares, respectively, in Figure 4.5(a). Then the grid join* $\mathcal{L}_1 \oplus \mathcal{L}_2$ *is generated by*

$$\mathcal{G}_1 \oplus \mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 3 & 0 & 2 & 0 \\ 2 & 4 & 2 & 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right);$$

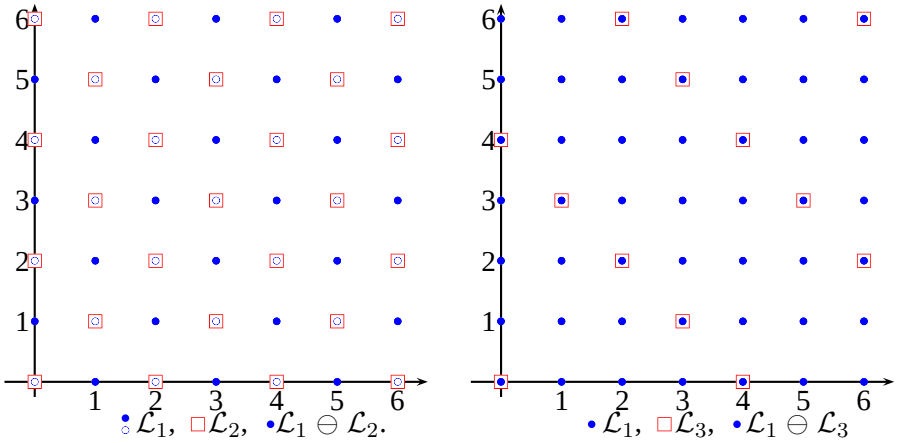
thus, the generator system

$$\mathcal{G} := \left(\varnothing, \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right)$$

is a minimal form of $\mathcal{G}_1 \oplus \mathcal{G}_2$ *and* $\mathcal{L}_1 \oplus \mathcal{L}_2 = \mathrm{ggen}(\mathcal{G})$*. The grid* $\mathcal{L}_1 \oplus \mathcal{L}_2$ *is illustrated by the filled squares in Figure 4.5(b). Note that here* $\mathcal{L}_1 \oplus \mathcal{L}_2 \neq \mathcal{L}_1 \cup \mathcal{L}_2$*.*

4.5 Difference

For any pair of grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *grid difference* of $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \ominus \mathcal{L}_2$, is defined as the smallest grid containing the set-theoretic difference of $\mathcal{L}_1$ and $\mathcal{L}_2$. A computation of grid difference is given in Example 4.9.



(a) Grid $\mathcal{L}_1 \ominus \mathcal{L}_2$.

(b) Grid $\mathcal{L}_1 \ominus \mathcal{L}_3$.

Figure 4.6: Grid difference.

Example 4.9 *Consider the three grids*

$$\mathcal{L}_1 := \mathrm{gcon}\big(\{x \equiv_1 0, y \equiv_1 0\}\big),$$
$$\mathcal{L}_2 := \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_2 0\}\big),$$
$$\mathcal{L}_3 := \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_4 0\}\big).$$

The grids $\mathcal{L}_1$ and $\mathcal{L}_2$ are illustrated by all the circles (open and filled) and open squares, respectively, in Figure 4.6(a). Then the grid difference

$$\mathcal{L}_1 \ominus \mathcal{L}_2 = \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_2 1\}\big)$$

is illustrated by the filled circles. The grids $\mathcal{L}_1$ and $\mathcal{L}_3$ are illustrated by the filled circles and open squares, respectively, in Figure 4.6(b). In this case, the grid difference is $\mathcal{L}_1 \ominus \mathcal{L}_3 = \mathcal{L}_1$, which is illustrated by the circles. Note that here $\mathcal{L}_1 \ominus \mathcal{L}_2 \neq \mathcal{L}_1 \setminus \mathcal{L}_2$.

We now introduce the algorithm that produces the grid difference. As we have seen in Example 4.9 the grid difference will only produce something other that $\mathcal{L}_1$ or $\varnothing$ if $\mathcal{L}_2$ divides the points of $\mathcal{L}_1$ exactly into two disjoint sets. Therefore informally the grid difference algorithm can be thought of as trying to split the points of the grid into a partition, so that each alternate point is in the other partition. This is only possible if the grid to be subtracted is equal to one of the partitions as can be seen in Figure 4.6(a). If $\mathcal{L}_1 = \mathcal{L}_2$ then $\varnothing$ is returned otherwise, in all other cases, $\mathcal{L}_1$ is returned.

Algorithm 1: The grid difference algorithm.
Input: Nonempty grids $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ in $\mathbb{G}_n$.
Output: A grid in $\mathbb{G}_n$.

(1) $\mathcal{L}' := \varnothing$
(2) **while** $\exists \beta = (e \equiv_f 0) \in \mathcal{C}_2$
(3) $\quad \mathcal{C}_2 := \mathcal{C}_2 \setminus \{\beta\}$
(4) $\quad$ **if** $\mathcal{L}_1 \nsubseteq \mathrm{gcon}\big(\{\beta\}\big)$
(5) $\quad\quad$ **if** $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{2e \equiv_f 0\}\big)$
(6) $\quad\quad\quad \mathcal{L}_\beta := \mathrm{gcon}\big(\mathcal{C}_1 \cup \{2e - f \equiv_{2f} 0\}\big)$
(7) $\quad\quad\quad \mathcal{L}' := \mathcal{L}' \oplus \mathcal{L}_\beta$
(8) $\quad\quad$ **else**
(9) $\quad\quad\quad$ **return** $\mathcal{L}_1$
(10) **return** $\mathcal{L}'$

Algorithm 1 provides an implementation for grid difference.

Proposition 4.10 *Let $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$ and suppose that $\mathcal{L}$ is the grid returned by* Algorithm 1. *Then $\mathcal{L} = \mathcal{L}_1 \ominus \mathcal{L}_2$.*

Proof. By the initial conditions, $\mathcal{L}_1 \neq \varnothing, \mathcal{L}_2 \neq \varnothing$. Let $\mathcal{L}'$ be the empty grid in $\mathbb{G}_n$ defined on line (1). Then the algorithm executes lines (2-10). Notice that there are two lines in this range that return a value for $\mathcal{L}$; line (9) when $\mathcal{L} = \mathcal{L}_1$ and line (10) when $\mathcal{L} = \mathcal{L}'$.

Consider first the case when line (9) is executed so that $\mathcal{L} = \mathcal{L}_1$. By definition of grid difference, $\mathcal{L} \supseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Hence it remains to show that $\mathcal{L}_1 \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. If $\mathbf{p} \in \mathcal{L}_1 \setminus \mathcal{L}_2$, then, by the definition of grid difference, $\mathbf{p} \in \mathcal{L}_1 \ominus \mathcal{L}_2$. Suppose now that $\mathbf{p} \in \mathcal{L}_1 \cap \mathcal{L}_2$. As line (9) is only executed by following the else branch of the conditional on line (5), for some congruence $\beta = (e \equiv_f 0) \in \mathcal{C}_2$, there exists a point $\mathbf{q} \in \mathcal{L}_1$ that does not satisfy $(2e \equiv_f 0)$ so that $\mathbf{q}$ does not satisfy β and hence $\mathbf{q} \notin \mathcal{L}_2$. Consider the point $\mathbf{r} = \mathbf{p} + 2(\mathbf{q} - \mathbf{p})$. Then, as $\mathbf{r}$ is an integral affine combination of points in $\mathcal{L}_1$, $\mathbf{r} \in \mathcal{L}_1$. Let $e = \big(\langle \mathbf{a}, \mathbf{x}\rangle - b\big)$. Then, as $\mathbf{p} \in \mathcal{L}_2$ satisfies β, $\big(\langle \mathbf{a}, \mathbf{p}\rangle - b \equiv_f 0\big)$. If $\mathbf{r}$ also satisfies β, then $\big(\langle \mathbf{a}, \mathbf{r}\rangle - b \equiv_f 0\big)$ and hence $\big(\langle \mathbf{a}, 2\mathbf{q}\rangle - 2b \equiv_f 0\big)$ so that $\mathbf{q}$ would satisfy $(2e \equiv_f 0)$; a contradiction. Thus $\mathbf{r} \notin \mathcal{L}_2$. Therefore $\mathbf{p} = 2\mathbf{q} - \mathbf{r}$ is an integral affine combination of points in $\mathcal{L}_1 \setminus \mathcal{L}_2$ and hence $\mathbf{p} \in \mathcal{L}_1 \ominus \mathcal{L}_2$. As $\mathbf{p} \in \mathcal{L}_1 = \mathcal{L}$ was arbitrary, $\mathcal{L} \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$.

Suppose now that line (9) is not executed. Then the loop iterates once for each congruence in $\mathcal{C}_2$ before executing line (10). Suppose $\#\,\mathcal{C}_2 = c$ and $\beta_i = (e_i \equiv_f 0) \in \mathcal{C}_2$ is the congruence selected at line (2) in the i-th iteration of the loop, for $0 < i \leq c$. Let $\mathcal{L}'_0 = \varnothing$ and $\mathcal{L}'_i$ denote the grid $\mathcal{L}'$ after the i-th iteration. Then we need to show that $\mathcal{L}'_c = \mathcal{L}_1 \ominus \mathcal{L}_2$. We prove that $\mathcal{L}'_c \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$ and $\mathcal{L}'_c \supseteq \mathcal{L}_1 \ominus \mathcal{L}_2$ separately.

We first show that $\mathcal{L}'_c \supseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Since $\mathcal{L}_1 \ominus \mathcal{L}_2$ is the smallest grid containing $\mathcal{L}_1 \setminus \mathcal{L}_2$, we just need to show that $\mathcal{L}'_c \supseteq \mathcal{L}_1 \setminus \mathcal{L}_2$. To do this, let $\mathbf{p} \in \mathcal{L}_1 \setminus \mathcal{L}_2$; then we prove that $\mathbf{p} \in \mathcal{L}'_c$. As $\mathbf{p} \notin \mathcal{L}_2$, for some $j = 1, \ldots, c$, $\mathbf{p} \notin \mathrm{gcon}\big(\{\beta_j\}\big)$. Consider the j-th iteration of the loop. Then

the test on line (4) will succeed and the execution continues with the test on line (5). Moreover, as we know that line (9) will not be executed, this test must succeed so that $\mathbf{p} \in \mathrm{gcon}\big(\{2e_j \equiv_f 0\}\big)$ and lines (6-7) will be executed with $\beta = \beta_j$. As $\mathrm{gcon}\big(\{\beta_j\}\big)$ and $\mathrm{gcon}\big(\{2e_j \equiv_f f\}\big)$ are disjoint and their set union is the grid $\mathrm{gcon}\big(\{2e_j \equiv_f 0\}\big)$, $\mathbf{p}$ must satisfy the congruence $\big(2e_j - f \equiv_{2f} 0\big)$. Let $\mathcal{L}_{\beta_j} = \mathrm{gcon}\big(\mathcal{C}_1 \cup \{2e_j - f \equiv_{2f} 0\}\big)$ as on line (6). Then, as $\mathbf{p} \in \mathcal{L}_1$, we have $\mathbf{p} \in \mathcal{L}_{\beta_j}$; hence, after line (7), $\mathbf{p} \in \mathcal{L}'_j$. For each $i = j+1, \ldots, c$, either $\mathcal{L}'_i = \mathcal{L}'_{i-1}$ or line (6) is executed, in which case $\mathcal{L}'_i \supseteq \mathcal{L}'_{i-1}$; hence $\mathbf{p} \in \mathcal{L}'_i$. In particular, $\mathbf{p} \in \mathcal{L}'_c$. As this holds for all $\mathbf{p} \in \mathcal{L}_1 \setminus \mathcal{L}_2$, $\mathcal{L}'_c \supseteq \mathcal{L}_1 \setminus \mathcal{L}_2$.

Finally we prove, by induction on i, that, for each $i = 0, \ldots, c$, $\mathcal{L}'_i \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Initially $\mathcal{L}'_0 = \varnothing$ and the result holds. Suppose now that $i > 0$ and that $\mathcal{L}'_{i-1} \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. If $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{\beta_i\}\big)$, then $\mathcal{L}'_i = \mathcal{L}'_{i-1}$ is unchanged by the iteration. On the other hand, if $\mathcal{L}_1 \not\subseteq \mathrm{gcon}\big(\{\beta_i\}\big)$, the test on line (4) will succeed and the execution continues with the test on line (5). Moreover, as we know that line (9) will not be executed, this test must succeed so that $\mathcal{L}_1 \subseteq \mathrm{gcon}\big(\{2e_i \equiv_f 0\}\big)$. Let $\mathcal{L}_{\beta_i} = \mathrm{gcon}\big(\mathcal{C}_1 \cup \{2e_i - f \equiv_{2f} 0\}\big)$ as defined on line (6); then $\mathcal{L}_{\beta_i} \cap \mathcal{L}_2 = \varnothing$ so that $\mathcal{L}_{\beta_i} \subseteq \mathcal{L}_1 \setminus \mathcal{L}_2 \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Since, on line (7), $\mathcal{L}'_i$ is assigned $\mathcal{L}'_{i-1} \oplus \mathcal{L}_{\beta_i}$, by definition of grid join and grid difference, $\mathcal{L}'_i \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. Therefore, letting $i = c$, we have $\mathcal{L}'_c \subseteq \mathcal{L}_1 \ominus \mathcal{L}_2$. □

Assuming $\mathcal{C}_1$ and $\mathcal{C}_2$ are known and in minimal form for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, it follows from the complexities of minimisation, conversion and comparison operations that the grid difference algorithm, Algorithm 1, has worst-case complexity $\mathrm{O}\big(n^4\big)$.

4.6 Rectilinear Grids

Recall from Definition 3.9 that a rectilinear grid is a grid that can be represented by a non-relational set of congruences or generators. In this section we will show how to compute, for any grid $\mathcal{L}$, the smallest rectilinear grid that contains $\mathcal{L}$. We also show how such grids can provide safe approximations for any rational grid. The following two propositions show that if we are given a grid represented by a generator system we can produce a rectilinear grid represented by either a congruence or generator system.

Proposition 4.11 *Let $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$ where $\mathcal{G} = \big(L, Q, \{\mathbf{p}\}\big)$. Let $q = \#\, Q$ and $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_q\}$. Let $\mathcal{L}' = \mathrm{gcon}(\mathcal{C}')$ such that:*

$$\mathcal{C}' := \Big\{ \big(\langle \mathbf{e}_i, \mathbf{x}\rangle \equiv_{|g|} p_i\big) \Big| 1 \le i \le n, \forall \boldsymbol{\ell} \in L : \ell_i = 0,\ g = \gcd\big(\{q_{1i}, \ldots, q_{qi}\}\big) \Big\}. \tag{4.3}$$

Then $\mathcal{L}'$ is the smallest rectilinear grid containing $\mathcal{L}$.

Proof. We first show that $\mathcal{L} \subseteq \mathcal{L}'$. Suppose that $\mathbf{v} \in \mathcal{L}$ and that for some $i \in \{1, \ldots, n\}$ and for all $\boldsymbol{\ell} \in L, \ell_i = 0$. Let $g = \gcd\big(\{q_{1i}, \ldots, q_{qi}\}\big)$, $\beta = \big(\langle \mathbf{e}_i, \mathbf{x}\rangle \equiv_{|g|} p_i\big)$, $l = \#\, L$ and $L = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_l\}$. As $\mathbf{v} \in \mathcal{L}$ we can assume that

$$\mathbf{v} = a_1 \cdot \boldsymbol{\ell}_1 + \cdots + a_l \cdot \boldsymbol{\ell}_l + b_1 \cdot \mathbf{q}_1 + \cdots + b_q \cdot \mathbf{q}_q + \mathbf{p},$$

for $a_1, \ldots, a_\ell \in \mathbb{R}$ and $b_1, \ldots, b_q \in \mathbb{Z}$. Thus $v_i = b_1 \cdot q_{1i} + \cdots + b_q \cdot q_{qi} + p$ and $v_i \equiv_{|g|} p_i$. Hence $\big(\langle \mathbf{e}_i, \mathbf{v}\rangle \equiv_{|g|} p_i\big)$, so $\mathbf{v}$ satisfies β. Hence for all i, where $1 \leq i \leq n$, $\mathbf{v}$ satisfies all the congruences of $\mathcal{C}'$ and therefore $\mathbf{v} \in \mathcal{L}'$.

Now to see that $\mathcal{L}'$ is the smallest rectilinear grid containing $\mathcal{L}$ let us suppose that there is another grid $\mathcal{L}''$ such that $\mathcal{L}''$ is rectilinear and $\mathcal{L} \subseteq \mathcal{L}'' \subseteq \mathcal{L}'$. Let $\mathcal{C}''$ be a congruence system such that $\mathcal{L}'' = \mathrm{gcon}(\mathcal{C}'')$. Suppose that, for some $i \in \{1, \ldots, n\}$ and $g'' > 0$ there is $\gamma = \big(\langle \mathbf{e}_i, \mathbf{x}\rangle \equiv_{|g''|} p_i\big) \in \mathcal{C}''$. Since $\mathcal{L} \subseteq \mathcal{L}''$, any line $\boldsymbol{\ell} \in L$ is also a line for a generator system that represents $\mathcal{L}''$. Thus, for all $a \in \mathbb{R}$, $\langle \mathbf{e}_i, a \cdot \boldsymbol{\ell}\rangle \equiv_{|g''|} 0$. Thus $\ell_i = 0$ and as this must hold for all $\boldsymbol{\ell} \in L$, there exists $\beta = \big(\langle \mathbf{e}_i, \mathbf{x}\rangle \equiv_{|g|} p_i\big) \in \mathcal{C}'$ where $g = \gcd\big(\{q_{1i}, \ldots, q_{qi}\}\big)$. Then, as $\mathcal{L} \subseteq \mathcal{L}''$, for all $1 \leq j \leq q$, $\mathbf{q}_j$ is a parameter of a generator system that represents $\mathcal{L}''$ so that $\langle \mathbf{e}_i, \mathbf{q}_j\rangle \equiv_{|g''|} 0$. Hence, for all $j \in \{1, \ldots, q\}$, $g''|q_{ji}$, so $g''|g$. Hence $\mathcal{L}' \subseteq \mathcal{L}''$, and as $\mathcal{L}'' \subseteq \mathcal{L}'$, that means $\mathcal{L}' = \mathcal{L}''$. Therefore $\mathcal{L}'$ is the smallest rectilinear grid that contains $\mathcal{L}$. □

Proposition 4.12 *Let $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$ where $\mathcal{G} = \big(L, Q, \{\mathbf{p}\}\big)$. Let $q = \#\, Q$ and $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_q\}$. Let $\mathcal{L}' = \mathrm{ggen}(\mathcal{G}')$ where $\mathcal{G}' = \big(L', Q', \{\mathbf{p}\}\big)$, such that, for each $i \in \{1, \ldots, n\}$:*

1. *if, for some $\boldsymbol{\ell} \in L$, $\ell_i \neq 0$, then let $\boldsymbol{\ell}'_{\boldsymbol{i}} := \mathbf{e}_i$;*
2. *if, for all $\boldsymbol{\ell} \in L$, $\ell_i = 0$, and for some $\mathbf{q}_j \in Q$, $q_{ji} \neq 0$ then let $\mathbf{q}'_i := |g| \cdot \mathbf{e}_i$ where $g = \gcd\big(\{q_{1i}, \ldots, q_{qi}\}\big)$.*

Then $\mathcal{L}'$ is the smallest rectilinear grid containing $\mathcal{L}$.

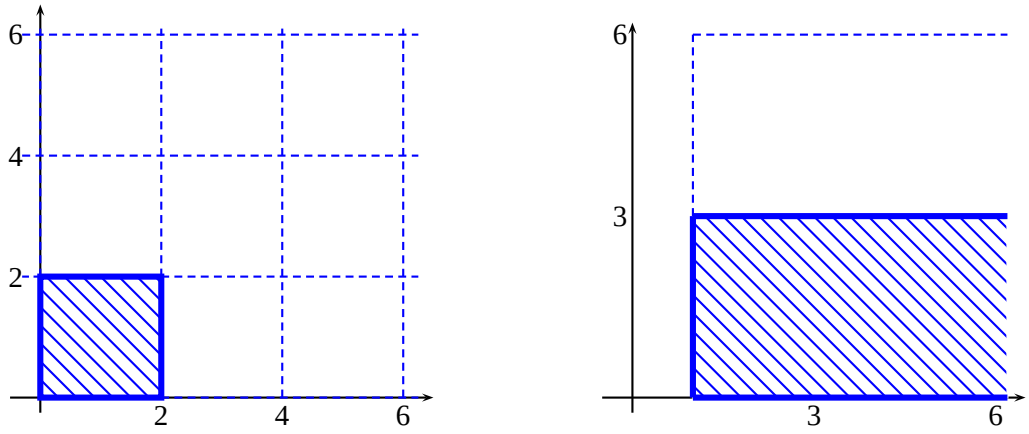
Proof. By Proposition 4.11 there is a grid $\mathcal{L}_1$ such that $\mathcal{L}_1$ is the smallest rectilinear grid that contains $\mathcal{L}$. Also by Proposition 4.11 there is a grid $\mathcal{L}_2$ such that $\mathcal{L}_2$ is the smallest rectilinear grid that contains $\mathcal{L}'$. Now by the definition of $\mathcal{L}'$ we have that $\mathcal{L}'$ is rectilinear thus $\mathcal{L}' = \mathcal{L}_2$. All that remains is to show that $\mathcal{L}' = \mathcal{L}_1$.

We will first show that $\mathcal{L}' \subseteq \mathcal{L}_1$. That is every generator of $\mathcal{G}'$ satisfies every congruence of $\mathcal{C}_1$. Suppose that for all $\boldsymbol{\ell} \in L$ there is some $i \in \{1, \ldots, n\}$ such that $\ell_i = 0$ and for some $j \in \{1, \ldots, q\}$, $\mathbf{q}_j \in Q$ and $q_{ji} \neq 0$. Then there is $\mathbf{q}'_i \in Q'$ such that $\mathbf{q}'_i = |g| \cdot \mathbf{e}_i$ and $\beta \in \mathcal{C}_1$ such that $\beta = \big(\langle \mathbf{e}_i, \mathbf{x}\rangle \equiv_{|g|} p_i\big)$, where $g = \gcd\big(\{q_{1i}, \ldots, q_{qi}\}\big)$. Hence, $\big(\langle \mathbf{e}_i, \mathbf{q}'_i\rangle \equiv_{|g|} 0\big)$. So $\mathbf{q}'_i$ satisfies β and therefore $\mathbf{q}'_i$ satisfies all congruences of $\mathcal{C}_1$. Hence all parameters of Q' satisfy each congruence of $\mathcal{C}_1$. Since $\mathcal{L} \subseteq \mathcal{L}_1$, any line $\boldsymbol{\ell} \in L$ is also a line for a generator system that represents $\mathcal{L}_1$. Thus, for all $a \in \mathbb{R}$, $\langle \mathbf{e}_i, a \cdot \boldsymbol{\ell}\rangle \equiv_{|g|} 0$. So if there is $\boldsymbol{\ell} \in L$ such that $\ell_i \neq 0$, then there is $\boldsymbol{\ell}'_{\boldsymbol{i}} \in L'$ such that $\boldsymbol{\ell}'_{\boldsymbol{i}} = \mathbf{e}_i$ and for all $a \in \mathbb{R}$, $\langle \mathbf{e}_i, a \cdot \boldsymbol{\ell}'_{\boldsymbol{i}}\rangle \equiv_{|g|} 0$. So $\mathcal{L}' \subseteq \mathcal{L}_1$.

Now as $\mathcal{L}_1$ is the smallest rectilinear grid that contains $\mathcal{L}$ we must have that $\mathcal{L}_1 \subseteq \mathcal{L}'$, hence $\mathcal{L}_1 = \mathcal{L}'$. □

4.6.1 Covering Box

In this section we will show how we can reuse the standard interval domain, see [26] and Chapter 2, to represent a *rectilinear grid*. Recall from Section 2.3.2 that we can represent a non-empty n-dimensional *rational box* $\mathcal{B}$ by a sequence $(I_1, \ldots, I_n)$ of rational intervals.



(a) A tiling using a bounded box.

(b) A tiling using an unbounded box.

Figure 4.7: Types of 2-dimensional box tilings.

An n-dimensional box $\mathcal{B}$ could be used repetitively over an n-dimensional vector space to "tile" and therefore "cover" the vector space. A tiling or tessellation of a vector space is a collection of objects that fill the vector space so that the objects do not overlap or leave gaps. Therefore it follows that the box $\mathcal{B}$ determines a *covering* of the n-dimensional vector space, where the given box $\mathcal{B}$ provides the position for one of the tiles. By defining a grid to be the vertices of the tiles in such a tiling, we obtain a rational rectilinear grid $\mathcal{L}$ and call $\mathcal{B}$ a *covering box for* $\mathcal{L}$. For this section we will assume that any unbounded interval has the form $[\mu, \infty]$. Example 4.13 shows informally how a box can tile a 2-dimensional vector space.

Example 4.13 *In Figure 4.7 two tilings are given, in Figure 4.7(a) both intervals are bounded and the box is given by* $\mathcal{B} = \big([0,2],[0,2]\big)$. *It can be seen that the box will tile the whole* $\mathbb{R}^2$ *vector space and the covering box will represent a rectilinear grid* $\mathcal{L}$ *such that* $\mathcal{L} = \mathrm{gcon}\big(\{x \equiv_2 0, y \equiv_2 0\}\big)$.

In Figure 4.7(b) only one interval is bounded and the box is given by $\mathcal{B} = \big([1,\infty],[0,3]\big)$. *It can be seen that the box will only tile the half-space* $\{\mathbf{x} \in \mathbb{R}^2 | 1 \leq x\}$. *This covering box will represent the grid* $\mathcal{L}$ *such that* $\mathcal{L} = \mathrm{gcon}\big(\{x = 1, y \equiv_3 0\}\big)$. *The equation* $x = 1$ *is approximated by an unbounded interval* $[1, \infty]$ *to show that the* x *variable only takes one value, so the tile is not repeated along the direction of the that variable.*

If the box $\mathcal{B} = (I_1, \ldots, I_n)$ has an interval I_i which is the singleton $[\mu, \mu]$ then if $\mathbf{v} \in \mathcal{L}$, $(v_1, \ldots, \lambda \cdot v_i, \ldots, v_n) \in \mathcal{L}$ for all $\lambda \in \mathbb{R}$. In other words, the generator representation for $\mathcal{L}$ contains a line. Informally, the reason a line is represented by a singleton is that the v_i variable will take all values λ, for $\lambda \in \mathbb{R}$. So a singleton is the smallest amount we can take before the next tile would occur. So in the direction of the line the tiling would be repeated infinitely many times, once for each $\lambda \in \mathbb{R}$.

Definition 4.14 (A Box Represents $\mathcal{L} = \mathbf{gcon}(\mathcal{C})$.) *Let* $\mathcal{B} = (I_1, \ldots, I_n)$ *be a non-empty box.*

For each $i = 1, \ldots, n$, let $I_i = [\mu_i, \nu_i]$; then, if $\mu_i \neq \nu_i$, let

$$\beta_i := \begin{cases} \big(\langle \mathbf{e}_i, \mathbf{x} \rangle \equiv_{\nu_i - \mu_i} \mu_i\big), & \text{if } I_i \text{ is bounded;} \\ \big(\langle \mathbf{e}_i, \mathbf{x} \rangle = \mu_i\big), & \text{if } I_i \text{ is not bounded.} \end{cases}$$

Then we say that the box $\mathcal{B}$ represents *the grid $\mathcal{L} := \mathrm{gcon}(\mathcal{C})$, where*

$$\mathcal{C} := \{\, \beta_i \;\big|\; 1 \leq i \leq n, \mu_i \neq \nu_i \,\}. \tag{4.4}$$

Note that the congruence system $\mathcal{C}$ is in minimal form. Observe also that, when $\mu_i = \nu_i$ for some $1 \leq i \leq n$, there is no corresponding congruence in $\mathcal{C}$ for $[\mu_i, \nu_i]$; this is because, in this case, the tiling will cover every value in this dimension and hence there will be a line $\mathbf{e}_i$ in the generator representation of $\mathcal{L}$.

Definition 4.15 (A Box Represents $\mathcal{L} = \mathbf{ggen}(\mathcal{G})$.) *Alternatively let $\mathcal{B} = (I_1, \ldots, I_n)$ be a non-empty box. For each $i = 1, \ldots, n$, let $I_i = [\mu_i, \nu_i]$; then let*

$$\mathbf{v}_i := \begin{cases} \mathbf{e}_i, & \text{if } \mu_i = \nu_i, \text{ so } \mathbf{v}_i \text{ is a line;} \\ |\nu_i - \mu_i| \cdot \mathbf{e}_i, & \text{if } \mu_i \neq \nu_i \text{ and } I_i \text{ is bounded, so } \mathbf{v}_i \text{ is a parameter;} \end{cases}$$

and

$$\mathbf{p} := (\mu_1, \ldots, \mu_n).$$

Then we say that the box $\mathcal{B}$ represents *the grid $\mathcal{L} := \mathrm{ggen}(\mathcal{G})$, where*

$$\mathcal{G} := \{\, \mathbf{v}_i \;\big|\; 1 \leq i \leq n \,\} \cup \{\mathbf{p}\}. \tag{4.5}$$

Note that the generator system $\mathcal{G}$ is in minimal form. Observe also that, when I_i is unbounded for some $1 \leq i \leq n$, there is a no generator in $\mathcal{G}$ for $[\mu_i, \nu_i]$; this is because, in this case, the tiling will cover only one value in this dimension and hence there will be an equation $\langle \mathbf{e}_i, \mathbf{x} \rangle = \mu_i$ in the congruence representation of $\mathcal{L}$.

Definition 4.16 (Covering Box.) *Let $\mathcal{L}$ be a non-empty rational grid. A* covering box *for $\mathcal{L}$ is a rational box representing the smallest rectilinear grid that contains $\mathcal{L}$.*

We now provide a procedure for computing the covering box of a grid.

Proposition 4.17 *Let $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$ where $\mathcal{G} = \big(L, Q, \{\mathbf{p}\}\big)$. Let $q = \#\, Q$ and $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_q\}$. Let $\mathcal{B} = (I_1, \ldots, I_n)$ such that, for each $i \in \{1, \ldots, n\}$:*

1. *if, for some $\boldsymbol{\ell} \in L$, $\ell_i \neq 0$, then let $I_i := [0, 0]$;*
2. *if, for all $\boldsymbol{\ell} \in L$, $\ell_i = 0$, and $q_{1i} = \cdots = q_{qi} = 0$, let $I_i := [p_i, \infty]$;*

3. *otherwise, let* $I_i := \big[p_i, p_i + |g|\big]$ *where* $g = \gcd\big(\{q_{1i}, \ldots, q_{qi}\}\big)$.

Then $\mathcal{B}$ *is a covering box for* $\mathcal{L}$.

Proof. By Proposition 4.11 and Proposition 4.12 we can compute the congruence system $\mathcal{C}'$ and generator system $\mathcal{G}'$, respectively, such that $\mathcal{L}' = \operatorname{gcon}(\mathcal{C}') = \operatorname{ggen}(\mathcal{G}')$ and $\mathcal{L}'$ is the smallest rectilinear grid containing $\mathcal{L}$. Then $\mathcal{C}'$ is the set given in Equation (4.3) and $\mathcal{G}' = \big(L', Q', \{\mathbf{p}\}\big)$ such that for each $i \in \{1, \ldots, n\}$ conditions (1) and (2) hold from Proposition 4.12. Let $\mathcal{B} = (I_1, \ldots, I_n)$ be the box. Then $\mathcal{G}'$ is equivalent to the system

$$\mathcal{G}'' := \{\, \mathbf{v}_i \;\big|\; 1 \leq i \leq n \,\} \cup \{\mathbf{p}\}$$

such that

$$\mathbf{v}_i := \begin{cases} \mathbf{e}_i, & \text{if } \mu_i = \nu_i, \text{ so } \mathbf{v}_i \text{ is a line;} \\ |\nu_i - \mu_i| \cdot \mathbf{e}_i, & \text{if } \mu_i \neq \nu_i \text{ and } I_i \text{ is bounded, so } \mathbf{v}_i \text{ is a parameter;} \end{cases}$$

and

$$\mathbf{p} := (\mu_1, \ldots, \mu_n)$$

and $\mathcal{C}'$ is equivalent to the system

$$\mathcal{C}'' := \{\, \beta_i \;\big|\; 1 \leq i \leq n, \mu_i \neq \nu_i \,\}$$

such that

$$\beta_i := \begin{cases} \big(\langle \mathbf{e}_i, \mathbf{x} \rangle \equiv_{\nu_i - \mu_i} \mu_i\big), & \text{if } I_i \text{ is bounded;} \\ \big(\langle \mathbf{e}_i, \mathbf{x} \rangle = \mu_i\big), & \text{if } I_i \text{ is not bounded.} \end{cases}$$

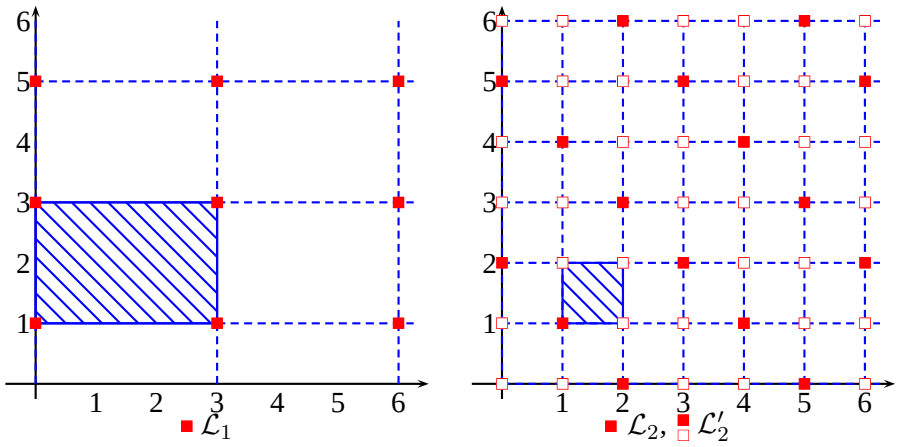
Then $\operatorname{gcon}(\mathcal{C}') = \operatorname{gcon}(\mathcal{C}'')$ and $\operatorname{ggen}(\mathcal{G}') = \operatorname{ggen}(\mathcal{G}'')$ and by Definitions 4.14 and 4.15, the box $\mathcal{B}$ represents the grid $\mathcal{L}' = \operatorname{gcon}(\mathcal{C}') = \operatorname{ggen}(\mathcal{G}')$. Hence $\mathcal{B}$ is a covering box for $\mathcal{L}$. $\square$

The complexity of computing the covering box for a grid $\mathcal{L} = \operatorname{ggen}(\mathcal{G})$ using the procedure given in Proposition 4.17, is $\mathrm{O}\big(nm\big)$, where $m = \#\,\mathcal{G}$. If however only the congruence system for $\mathcal{L}$ is known then the complexity of computing the covering box is that of minimisation and conversion, namely, $\mathrm{O}\big(mn \min\{m, n\}\big)$ where $m = \#\,\mathcal{C}$. Two computations of the covering box are given in Example 4.18.

Example 4.18 *Consider grid* $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1) = \operatorname{ggen}(\mathcal{G}_1)$, *where* $\mathcal{C}_1 := \{x \equiv_3 0, y \equiv_2 1\}$ *and*

$$\mathcal{G}_1 := \left(\varnothing, \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right).$$

The grid $\mathcal{L}_1$ *is illustrated by all the squares in Figure 4.8(a) and it can be seen that* $\mathcal{L}_1$ *is rectilinear and and box* $\mathcal{B}_1 = \big\{[0, 3], [1, 3]\big\}$ *is a covering box representing* $\mathcal{L}_1$. *The box* $\mathcal{B}_1$ *is illustrated by the hatched area in Figure 4.8(a) and the covering is represented by the dashed lines.*



(a) Grid $\mathcal{L}_1$. (b) Grids $\mathcal{L}_2$ and $\mathcal{L}'_2$.

Figure 4.8: Covering boxes for a grid.

Consider the grid $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{ggen}(\mathcal{G}_2)$, *where* $\mathcal{C}_2 := \{x \equiv_1 0, x + y \equiv_3 2\}$ *and*

$$\mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 1 & 0 \\ -1 & 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}\right).$$

The grid $\mathcal{L}_2$ *is illustrated by all the filled squares in Figure 4.8(b) and it can be seen that* $\mathcal{L}_2$ *is not rectilinear and that the box* $\mathcal{B}_2 = \big\{[1,2],[1,2]\big\}$, *is a covering box for* $\mathcal{L}_2$. *Thus* $\mathcal{B}_2$ *represents the grid* $\mathcal{L}'_2 = \mathrm{gcon}\big(\{x \equiv_1 1, y \equiv_1 1\}\big)$ *which is illustrated by all the squares (open and filled) in Figure 4.8(b). The box* $\mathcal{B}_2$ *is illustrated by the hatched area in Figure 4.8(b) and the covering is represented by the dashed lines.*

Note that, in general, a grid $\mathcal{L} \in \mathbb{G}_n$ does not have a unique covering box. For instance, if $\mathcal{B} = (I_1, \ldots, I_i, \ldots, I_n)$ is a covering box for $\mathcal{L}$, and the interval $I_i = [\mu_i, \nu_i]$ is bounded, then the box $\mathcal{B}' = (I_1, \ldots, I'_i, \ldots, I_n)$ is also a covering box for $\mathcal{L}$ if, for some $m \in \mathbb{Z}$, $I'_i = \big[\mu_i + m(\nu_i - \mu_i), \nu_i + m(\nu_i - \mu_i)\big]$. Example 4.19 illustrates this.

Example 4.19 *Recall from Example 4.18 and Figure 4.8(b) the grid* $\mathcal{L}_2$ *where* $\mathcal{L}_2 = \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_3 2\}\big)$ *and* $\mathcal{B}_2 = \big\{[1,2],[1,2]\big\}$. *Then it can be seen that* $\mathcal{B}'_2 = \big\{[0,1],[0,1]\big\}$ *and* $\mathcal{B}''_2 = \big\{[4,5],[2,3]\big\}$ *are also covering boxes for* $\mathcal{L}_2$.

Although the covering box is not unique it could be enforced by computing it from a grid which is represented by a homogeneous system in strong minimal form.

4.7 Affine Image and Pre-image

An affine transformation on the vector space $\mathbb{R}^n$ is a transformation which preserves collinearity and ratios of distances. That is an affine transformation takes points along a line and maps them to points along a line, maps a midpoint of a line segment to a midpoint and preserves intersection

```
q♯(L)
  if ...  then
    L := φ(L, u, 3u)
    L := φ(L, v, u + v)
    q♯(L)
    L := φ(L, u, 5u)
    L := φ(L, v, u + v)
  endif
```

Figure 4.9: An abstraction of q.

properties between lines. However, it does not preserve the angles or lengths of lines. Affine transformations can be represented by matrices in $\mathbb{R}^{n\times n}$ and it follows that the set $\mathbb{G}_n$ is closed under the set of all affine transformations for $\mathbb{R}^n$. The affine image and affine pre-image operators are provided by a 'single update'. Given a grid $\mathcal{L} \in \mathbb{G}_n$, a variable x_k and linear expression $e = \langle \mathbf{a}, \mathbf{x}\rangle + b$ with coefficients in $\mathbb{Q}$, the *affine image operator* $\phi(\mathcal{L}, x_k, e)$ maps the grid $\mathcal{L}$ to

$$\Big\{ \big(p_1, \ldots, p_{k-1}, \langle \mathbf{a}, \mathbf{p}\rangle + b, p_{k+1}, \ldots, p_n\big)^{\mathrm{T}} \in \mathbb{R}^n \;\Big|\; \mathbf{p} \in \mathcal{L} \Big\}.$$

Conversely, the *affine pre-image operator* $\phi^{-1}(\mathcal{L}, x_k, e)$ maps the grid $\mathcal{L}$ to

$$\Big\{ \mathbf{p} \in \mathbb{R}^n \;\Big|\; \big(p_1, \ldots, p_{k-1}, \langle \mathbf{a}, \mathbf{p}\rangle + b, p_{k+1}, \ldots, p_n\big)^{\mathrm{T}} \in \mathcal{L} \Big\}.$$

Observe that the affine image $\phi(\mathcal{L}, x_k, e)$ and pre-image $\phi^{-1}(\mathcal{L}, x_k, e)$ are invertible if and only if the coefficient a_k in the vector $\mathbf{a}$ is non-zero.

Example 4.20 *Consider again Example 3.12 and the recursive procedure in Figure 3.4. Taking the initial grid to be $\mathcal{L}_0$, then* `x := 3*x`*, the first assignment in* `q`*, corresponds to the transformation $\phi(\mathcal{L}_0, u, 3u)$. This returns $\mathcal{L}_0^1 := \mathrm{ggen}(L, \varnothing, P_0^1)$ where $P_0^1 = \{(3,0,0)^{\mathrm{T}}\}$. The next assignment in* `q` *is* `y := x + y`*. The corresponding affine transformation, applied to $\mathcal{L}_0^1$, is $\phi(\mathcal{L}_0^1, v, v+u)$ and we obtain the grid $\mathcal{L}_0^2 := \mathrm{ggen}(L, \varnothing, P_0^2)$ where $P_0^2 = \{(3,3,0)^{\mathrm{T}}\}$. Now the assignment in* `q` *is* `y := 5*x`*. The corresponding affine transformation, applied to $\mathcal{L}_0^2$, is $\phi(\mathcal{L}_0^2, v, 5u)$ and we obtain the grid $\mathcal{L}_0^3 := \mathrm{ggen}(L, \varnothing, P_0^3)$ where $P_0^3 = \{(15,3,0)^{\mathrm{T}}\}$. Finally the last assignment in* `q` *is* `y := x + y`*. The corresponding affine transformation, applied to $\mathcal{L}_0^3$, is $\phi(\mathcal{L}_0^3, v, v+u)$ and we obtain the grid $\mathcal{L}_0^4 := \mathrm{ggen}(L, \varnothing, P_0^4)$ where $P_0^4 = \{(15,18,0)^{\mathrm{T}}\}$. Figure 4.9 contains an abstract version $\mathsf{q}^\sharp$ of* `q` *where the the argument to the procedure is replaced by a grid and the assignment statements are replaced by the corresponding affine transformations. Thus we can now compute the grids $\mathcal{L}_i = \mathrm{ggen}(L, \varnothing, P_i)$ for any i where i is the number of iterations through the body of the procedure. In particular $P_1 = P_0^4$ and we have computed P_0, P_1, P_2 and P_3, as seen in Example 3.12. Hence $\mathcal{L} = \mathcal{L}_0 \oplus \mathcal{L}_1 \oplus \mathcal{L}_2 \oplus \mathcal{L}_3$ represents a*

```
p(var u, var v)
  u := 3u + 2v + 1
  while ...
    u := u + 3
  endwhile
```

Figure 4.10: A simple procedure.

fixpoint for the procedure, thus it includes all possible values for the vector $(x, y)^{\mathrm{T}}$ *that might be obtained as a result of calling* q. *Then if we call the procedure with the values* $x = 2$ *and* $y = 0$ *as in [61], then all the possible values for the vector* $(x, y)^{\mathrm{T}}$ *are represented by the grid*

$$\mathrm{gcon}\big(\{x \equiv_{28} 2, y \equiv_{12} 0\}\big) = \phi\big(\phi(\mathcal{L}, x, 2x), y, 2y\big).$$

We will now introduce the generalized affine image operator. This determines a set of congruence relations that hold between the given grid and its image. Clearly, since the relations are congruences, the image is also a grid. Note though, that in this case a hyperplane will be replaced by a, possibly infinite, set of hyperplanes.

The *generalized affine image* (resp., *generalized affine pre-image*) is an extension of the affine image (resp., affine pre-image) operator defined above. Given a grid $\mathcal{L} \in \mathbb{G}_n$, linear expressions $e' = \langle \mathbf{c}, \mathbf{x} \rangle + d$ and $e = \langle \mathbf{a}, \mathbf{x} \rangle + b$ with coefficients in $\mathbb{Q}$ and $f \in \mathbb{Q}$, the generalized affine image operator $\psi = \psi(\mathcal{L}, e', e, f)$ is defined as

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^n : (\mathbf{v}, \mathbf{w}) \in \psi \iff \big(\langle \mathbf{c}, \mathbf{w} \rangle + d \equiv_f \langle \mathbf{a}, \mathbf{v} \rangle + b\big) \wedge \Big(\bigwedge_{\substack{0 \le i < n \\ c_i = 0}} w_i = v_i \Big).$$

Note that, when $e' = x_k$ and $f = 0$, then the transformation is equivalent to the standard affine transformation on $\mathcal{L}$ with respect to the variable x_k and the affine expression e; that is

$$\psi(\mathcal{L}, x_k, e, 0) = \phi(\mathcal{L}, x_k, e).$$

However, when $e' = x_k$ and $f = 1$, then the transformation maps the point $\mathbf{x}$ to the set of points $\big\{ (x_1, \ldots, x_{k-1}, x'_k, x_{k+1}, \ldots, x_n) \;\big|\; x'_k \equiv_1 \langle \mathbf{a}, \mathbf{x} \rangle + b \big\}$.

The following example illustrates how the generalized affine image can be used to model the effect of a procedure containing a while loop.

Example 4.21 *Consider the program procedure in Figure 4.10. Suppose the procedure takes the initial values* (a, b). *Then the procedure* p *will map* u *to* $3a + 2b + 1 + 3i$ *for some* $i \in \mathbb{Z}$ *and leave* $\mathsf{v} = b$ *unchanged. Now considering the general case, where the initial values for* u *and* v *are given by the points of the grid* $\mathcal{L}$, *then, after executing* p, *the values of* u *and* v *will be given*

by the points of the grid $\psi(\mathcal{L}, x, 3x + 2y + 1, 3)$*. Let us now consider the specific initial condition* $\mathcal{L} = \mathrm{ggen}(\varnothing, \varnothing, P)$ *where* $P = \left(\begin{smallmatrix} 0 & 3 & 0 \\ 0 & 0 & 3 \end{smallmatrix}\right)$*. By considering one point* $\mathbf{p} := \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)$ *we can see that when* ψ *is applied to the point* $\mathbf{p}$ *we get the set*

$$\left\{ \ldots, \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 7 \\ 0 \end{pmatrix}, \ldots \right\}.$$

Hence,

$$\psi(\mathcal{L}, x, 3x + 2y + 1, 3) = \mathrm{ggen}\left(\varnothing, \varnothing, \begin{pmatrix} 1 & 4 & 1 \\ 0 & 0 & 3 \end{pmatrix}\right).$$

4.8 Implementation

The intersection and grid join just take the union of the congruence or generator systems, respectively, so that, from a theoretical perspective, these have complexity $\mathrm{O}(n)$ as noted in Sections 4.3 and 4.4, respectively. However, in the implementation, we assume a common divisor for all the coordinates or coefficients in the system. Hence, combining the systems requires changing the denominators of both components to their least common multiple with a consequential need to scale all the numerators in the representation; giving a worst-case complexity of $\mathrm{O}(n^2)$ if both systems are in minimal form.

There are many operations that a practical domain of grids could provide for applications in program analysis and verification. For instance, the *concatenation* of two grids $\mathcal{L}_1 \in \mathbb{G}_n$ and $\mathcal{L}_2 \in \mathbb{G}_m$ (taken in this order) is the grid in $\mathbb{G}_{n+m}$ defined as

$$\left\{ (x_1, \ldots, x_n, y_1, \ldots, y_m)^{\mathrm{T}} \in \mathbb{R}^{n+m} \;\middle|\; \begin{array}{l} (x_1, \ldots, x_n)^{\mathrm{T}} \in \mathcal{L}_1 \\ (y_1, \ldots, y_m)^{\mathrm{T}} \in \mathcal{L}_2 \end{array} \right\}.$$

Other operators that could be required are those which that add, remove, rename and map the space dimensions, or expand and fold them along the lines of [36]. All of these operations have been specified and implemented, within the Parma Polyhedra Library, `http://www.cs.unipr.it/ppl/`, where all the code and documentation is publicly available.

4.9 Related Work

In [38,39] Granger considers operators for comparing grids and computing the greatest lower and least upper bounds, that is the intersection and join respectively. In particular in [39, Section 7] the complexities of each of the operations is stated. The join operation has complexity $\mathrm{O}(n^4 \log_2 n)$, this is because the operation takes generators of one grid and adds them one at a time to the generators of the other grid and at each stage minimises this new system. Unfortunately, as Granger's generator minimisation algorithm has complexity $\mathrm{O}(n^3 \log_2 n)$, it is this repeated minimisation that causes the complexity of the join to be so high. If our grid join operation where to be applied

similarly to a set of n generators we would have a complexity of $\mathrm{O}(n^4)$. The grid meet operation which also minimises the addition of one congruence at a time has complexity $\mathrm{O}(n^4)$ as the congruence minimisation has complexity $\mathrm{O}(n^3)$ and it is performed at worst n times. Finally the comparison has a complexity of $\mathrm{O}(n^3)$.

The operations provided by Quinton et al. [71, 72] for the grid part of the $\mathbb{Z}$-polyhedra which are similar to our operations are those grid intersection, affine image and affine pre-image. The operations of grid join and grid difference, where the result is a single grid, are not considered. Instead the join operator takes two grids $\mathcal{L}_1$ and $\mathcal{L}_2$ and returns the set $\{\mathcal{L}_1, \mathcal{L}_2\}$ unless one, say $\mathcal{L}_1$, is contained in the other, in which case they return the larger, $\mathcal{L}_2$. Similarly the difference operation returns a set of lattices representing the set difference $\mathcal{L}_1 \setminus \mathcal{L}_2$. This is calculated by computing a basis of parameters for each of the two lattices using the Smith Normal Form algorithm [67]. So if $\mathcal{L}_1 = \mathrm{ggen}(\varnothing, Q_1, \{\mathbf{p}_1\})$ where $Q_1 = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$ is the basis of parameters, then $\mathcal{L}_2 = \mathrm{ggen}(\varnothing, Q_2, \{\mathbf{p}_2\})$ where $Q_2 = (a_1\mathbf{q}_1, \ldots, a_n\mathbf{q}_n)$. Then the set theoretic difference is

$$\mathcal{L}_1 \setminus \mathcal{L}_2 := \Big(\bigcup_{i=1}^{m} \mathrm{ggen}(\varnothing, Q_2, \{\mathbf{v}_i\})\Big) \setminus \mathcal{L}_1 \cap \mathcal{L}_2$$

where the points $\mathbf{v}_i$ for each of the distinct grids to be in the union are

$$\mathbf{v}_i := \mathbf{p}_1 + \sum_{j=1}^{n} \sum_{k=0}^{a_j - 1} k\mathbf{q}_j$$

for $1 \leq i \leq m$ where $m := \prod_{j=1}^{n} a_j$. Therefore the overall complexity of calculating the difference is $\mathrm{O}(n^2 m)$. As there is no congruence representation, the intersection of two lattices is computed directly from the generator representations [1]; a refined version of this method is provided in [71] which we note that, as for computing the canonic form, has complexity $\mathrm{O}(n^4)$.

Muller-Olm and Seidl [60–62] consider analysing programs whose basic statements are either affine assignments or non-deterministic assignments. The affine assignments are therefore affine transformations on a single variable at a time, equivalent to that described here. The join is computed by adding an extra generator to a system that is already in minimal form. Thus at each stage the minimisation algorithm must be applied. Unfortunately, like Granger, to join two systems of n generators requires applying their minimisation algorithm n times, thus giving the join a complexity of $\mathrm{O}(n^4 \log_2 n)$.

4.10 Conclusion

We have shown in this chapter that we have operations for the domain that approximate the set-theoretic operations by producing a single grid. We introduced intersection and join operations with worst case complexity $\mathrm{O}(n^3)$ for both, which improves on previous proposals. We described a grid difference operator which returns the smallest single grid that contains the set-theoretic difference. We proposed a single update affine image and pre-image operators as well as new

generalised affine image and pre-image operators which maps single points to sets of points. Finally we have introduced the notion of a rectilinear grid, that is, a grid that can be represented by a set of non-relational congruences. Then we have shown that we can reuse the interval domain by creating a covering box for a grid $\mathcal{L}$, that is, a box that represents the smallest rectilinear grid that contains $\mathcal{L}$.

Chapter 5

Grid Widening and Weakly Relational Grids

5.1 Grid Widening

It was observed by Granger [41] that, if the grid generators can be in the rationals, then the grid domain does not satisfy the ascending chain condition.

Example 5.1 *Consider the grids $\mathcal{L}_i = \mathrm{ggen}(\mathcal{G}_i)$ for $i \in \mathbb{Z}$, where*

$$\mathcal{G}_i := \left(\varnothing, \begin{pmatrix} \frac{1}{2^i} & 0 \\ 0 & \frac{1}{2^i} \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right).$$

Then for each i, $\mathcal{L}_i \subseteq \mathcal{L}_{i+1}$. Hence we have an infinite increasing chain.

So to guarantee termination of the analysis, a widening operation is required. A simple and general characterization of a widening for enforcing and accelerating convergence of an upward iteration sequence is given in [26, 27, 30, 31]. We assume here a minor variation of this classical definition (see footnote 6 in [31, Page 275]).

Definition 5.2 (Widening.) *Let $\langle D, \vdash, \mathbf{0}, \oplus \rangle$ be a join-semilattice. The partial operator $\nabla \colon D \times D \rightarrowtail D$ is a* widening *if*

1. *for each $d_1, d_2 \in D$, $d_1 \vdash d_2$ implies that $d_1 \nabla d_2$ is defined and $d_2 \vdash d_1 \nabla d_2$;*
2. *for each increasing chain $d_0 \vdash d_1 \vdash \cdots$, the increasing chain defined by $d'_0 := d_0$ and $d'_{i+1} := d'_i \nabla (d'_i \oplus d_{i+1})$, for $i \in \mathbb{N}$, is not strictly increasing.*

Example 5.3 *Let us consider the simple piece of code*

```
x := 8                          (P1)
for i := 1 to m                 (P2)
    x := x/2                    (P3)
endfor
```

Let $\mathcal{L}_j^i \in \mathbb{G}_2$ *denote the grid computed at the* i*-th iteration executed by the point* `P`j*. Initially,* $\mathcal{L}_j^0 = \varnothing = \mathrm{gcon}\big(\{1 = 0\}\big)$*, for* $j = 1, \ldots, 3$*. After the first iteration of the loop we have the following grids:*

$$\begin{aligned}\mathcal{L}_1^1 &= \mathrm{gcon}\big(\{x = 8\}\big),\\ \mathcal{L}_2^1 &= \mathrm{gcon}\big(\{x = 8\}\big),\\ \mathcal{L}_3^1 &= \mathrm{gcon}\big(\{x = 4\}\big).\end{aligned}$$

Then after the second iteration of the loop we have the grids:

$$\begin{aligned}\mathcal{L}_2^2 &= \mathrm{gcon}\big(\{x = 8\}\big) \oplus \mathrm{gcon}\big(\{x = 4\}\big)\\ &= \mathrm{gcon}\big(\{x \equiv_4 0\}\big),\\ \mathcal{L}_3^2 &= \mathrm{gcon}\big(\{x \equiv_2 0\}\big).\end{aligned}$$

Then after the third iteration of the loop we have the grids:

$$\begin{aligned}\mathcal{L}_2^3 &= \mathrm{gcon}\big(\{x \equiv_4 0\}\big) \oplus \mathrm{gcon}\big(\{x \equiv_2 0\}\big)\\ &= \mathrm{gcon}\big(\{x \equiv_2 0\}\big),\\ \mathcal{L}_3^3 &= \mathrm{gcon}\big(\{x \equiv_1 0\}\big).\end{aligned}$$

It can be seen that after i *iterations we will have the grid* $\mathcal{L}_3^i = \mathrm{gcon}\big(\{x \equiv_{4/i} 0\}\big)$*. In this case the widening would be used an acceleration tool that can approximate the grid we would have at the end of the* `for` *loop without having to calculate each iteration.*

As well as formal requirements in Definition 5.2 given above, we also believe it is important to require that, as with all operations, we have a widening that has an efficient implementation. We will give two widenings, one for grids represented by a congruence system and one for grids represented by a generator system. Both of the widenings will assume that one of the grids is represented by the congruence or generator system in strong minimal form. We will show that these widenings are well defined and come with simple syntactic checks which have an efficient implementation.

5.2 Congruence Representation Widening

We introduce below the widening that is performed on grids which are represented by a congruence system. The widening also assumes that one of the grids is represented in minimal form and the other is represented in strong minimal form.

Definition 5.4 (Grid Widening for the Congruence System.) *Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ be two grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{C}_1$ is in minimal form and $\mathcal{C}_2$ is in strong minimal form. Then the* grid widening $\mathcal{L}_1 \nabla_c \mathcal{L}_2$ *is defined by*

$$\mathcal{L}_1 \nabla_c \mathcal{L}_2 := \begin{cases} \mathcal{L}_2, & \text{if } \mathcal{L}_1 = \varnothing \text{ or } \dim(\mathcal{L}_1) < \dim(\mathcal{L}_2), \\ \mathrm{gcon}(\mathcal{C}_S), & \text{otherwise,} \end{cases}$$

where $\mathcal{C}_S := \{\, \gamma \in \mathcal{C}_2 \mid \exists \beta \in \mathcal{C}_1 \,.\, \beta \Uparrow \gamma \,\}$.

The following proposition will show that '∇_c' satisfies the conditions of Definition 5.2 and therefore that it is a widening.

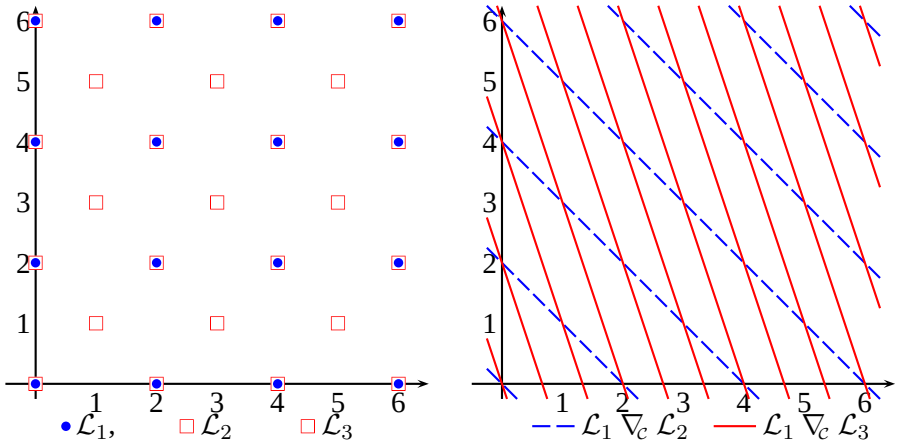
Proposition 5.5 *The operator '∇_c' is a widening on $\mathbb{G}_n$.*

Proof. In order to show that '∇_c' is a widening operator, we prove that conditions (1) and (2) in Definition 5.2 hold. Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) \in \mathbb{G}_n$, where $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{C}_1$ is in minimal form and $\mathcal{C}_2$ is in strong minimal form.

By Definition 5.4, if $\mathcal{L}_1 = \varnothing$ or $\dim(\mathcal{L}_1) < \dim(\mathcal{L}_2)$, then $\mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathcal{L}_2$. Therefore, in this case, condition (1) holds. Clearly, the empty grid can occur only as the first element of a strictly increasing chain of grids; moreover, if $\mathcal{L}$ and $\mathcal{L}'$ are any two successive and distinct grids in the increasing chain of condition (2) in Definition 5.2, then $0 \leq \dim(\mathcal{L}) \leq \dim(\mathcal{L}') \leq n$. Hence, the case when $\mathcal{L}_1 = \varnothing$ or $\dim(\mathcal{L}_1) < \dim(\mathcal{L}_2)$ hold can occur no more than a finite number of times in such a chain.

Suppose now that $\mathcal{L}_1 \neq \varnothing$ and $\dim(\mathcal{L}_1) = \dim(\mathcal{L}_2)$, so that the second case of the widening computation applies (note that, due to the inclusion hypothesis, $\dim(\mathcal{L}_1) > \dim(\mathcal{L}_2)$ cannot hold), and let $\mathcal{C}_S$ be as given in Definition 5.4. Then, since $\mathcal{C}_S \subseteq \mathcal{C}_2$, condition (1) holds. By Proposition 4.2, if $\mathcal{C}_S = \mathcal{C}_2$, we have $\mathcal{L}_1 = \mathcal{L}_2$; thus, if $\mathcal{L}_1 \neq \mathcal{L}_2$, we have $\#\mathcal{C}_S < \#\mathcal{C}_2$. By Lemma 3.34, as $\mathcal{C}_1$ and $\mathcal{C}_2$ are in minimal form, it follows that $\#\mathcal{C}_2 \leq \#\mathcal{C}_1$ so that, if $\mathcal{L}_1 \neq \mathcal{L}_2$, $\#\mathcal{C}_S < \#\mathcal{C}_1$. Therefore condition (2) of Definition 5.2 holds. □

Assuming that $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ and we know $\mathcal{L}_1 \subseteq \mathcal{L}_2$ this widening can be implemented to have complexity $\mathrm{O}(n^2)$, this is since all that is required is the copying of at most n congruences from $\mathcal{L}_2$ to $\mathcal{L}_1 \nabla_c \mathcal{L}_2$. If however the congruence system $\mathcal{C}_1$ is not in minimal form and the system $\mathcal{C}_2$ is not in strong minimal form, then the complexity of widening is that of the minimisations, namely $\mathrm{O}(mn \min\{m,n\})$, where $\#\mathcal{C}_1 = m_1, \#\mathcal{C}_2 = m_2$ and $m = \max\{m_1, m_2\}$. In Definition 5.4, it is required that $\mathcal{C}_2$ is in strong minimal form. Example 5.6 shows that this is necessary for the operator '∇_c' to be well-defined.



(a) Grids $\mathcal{L}_1$ and $\mathcal{L}_2$. (b) Grids $\mathcal{L}_1 \nabla_c \mathcal{L}_2$ and $\mathcal{L}_1 \nabla_c \mathcal{L}_3$.

Figure 5.1: Grid Widening.

Example 5.6 *Let* $\mathcal{L}_1 := \mathrm{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 := \mathrm{gcon}(\mathcal{C}_2)$ *and* $\mathcal{L}_3 := \mathrm{gcon}(\mathcal{C}_3)$ *where*

$$\begin{aligned}\mathcal{C}_1 &= \{x \equiv_2 0,\ y \equiv_2 0\},\\ \mathcal{C}_2 &= \{x \equiv_1 0,\ x + y \equiv_2 0\},\\ \mathcal{C}_3 &= \{x \equiv_1 0,\ 3x + y \equiv_2 0\};\end{aligned}$$

then $\mathcal{L}_2 = \mathcal{L}_3$. *The grid* $\mathcal{L}_1$ *is illustrated in Figure 5.1(a) by the filled circles and the grids* $\mathcal{L}_2, \mathcal{L}_3$ *are illustrated in Figure 5.1(a) by the open squares. Note that only* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are in strong minimal form. Therefore, assuming* $\mathcal{C}_S$ *(resp.,* $\mathcal{C}_S{}'$*) is defined as in Definition 5.4 using* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *(resp.,* $\mathcal{C}_1$ *and* $\mathcal{C}_3$*), we have*

$$\mathcal{C}_S = \{x + y \equiv_2 0\} \quad \textit{and} \quad \mathcal{C}_S{}' = \{3x + y \equiv_2 0\}.$$

The grid $\mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_S)$ *is illustrated in Figure 5.1(b) by the dashed lines and the grid* $\mathcal{L}_1 \nabla_c \mathcal{L}_3 = \mathrm{gcon}(\mathcal{C}_S{}')$ *is illustrated in Figure 5.1(b) by the complete lines. Thus* $\mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_S) \neq \mathrm{gcon}(\mathcal{C}_S{}')$.

The following example shows that the result of applying the widening ∇_c depends on the variable ordering.

Example 5.7 *To see that the widenings depend on the variable ordering, consider the grids* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1) = \mathrm{gcon}(\mathcal{C}'_1)$ *and* $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{gcon}(\mathcal{C}'_2)$ *in* $\mathbb{G}_2$, *where*

$$\begin{aligned}\mathcal{C}_1 &:= \{5x + y \equiv_1 0,\ 22x \equiv_1 0\}, \quad \mathcal{C}_2 := \{5x + y \equiv_1 0,\ 44x \equiv_1 0\},\\ \mathcal{C}'_1 &:= \{9y + x \equiv_1 0,\ 22y \equiv_1 0\}, \quad \mathcal{C}'_2 := \{9y + x \equiv_1 0,\ 44y \equiv_1 0\}.\end{aligned}$$

Assume for $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *that the variables are ordered so that* x *precedes* y, *as in the vector* $(x, y)^{\mathrm{T}}$;

then, $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form and, according to Definition 5.4, we obtain

$$\mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathrm{gcon}\big(\{5x + y \equiv_1 0\}\big).$$

On the other hand, $\mathcal{C}'_1$ and $\mathcal{C}'_2$ are in strong minimal form when taking the variable order where y precedes x. In this case, by Definition 5.4,

$$\mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathrm{gcon}\big(\{9y + x \equiv_1 0\}\big).$$

5.3 Generator Representation Widening

We introduce below the widening that is performed on grids which are represented by a generator system. The widening also assumes that one of the grids is represented in minimal form and the other is represented in strong minimal form.

Definition 5.8 (Grid Widening for the Generator System.) *Let $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$ be two grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{G}_1 = (L_1, Q_1, P_1)$ is in minimal form and $\mathcal{G}_2 = (L_2, Q_2, P_2)$ is in strong minimal form. Then the* grid widening *$\mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2$ is defined by*

$$\mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2 := \begin{cases} \mathcal{L}_2, & \text{if } \mathcal{L}_1 = \varnothing \text{ or } \dim(\mathcal{L}_1) < \dim(\mathcal{L}_2); \\ \mathrm{ggen}(\mathcal{G}_S), & \text{otherwise,} \end{cases}$$

where $\mathcal{G}_S := \big(L_2 \cup (Q_2 \setminus Q_S), Q_S, P_2\big)$ and $Q_S := \{\, \mathbf{v} \in Q_2 \mid \exists \mathbf{u} \in Q_1 \,.\, \mathbf{u} \Downarrow \mathbf{v} \,\}$.

The following proposition will show that '$\nabla_{\mathcal{G}}$' satisfies the conditions of Definition 5.2 and therefore that it is a widening.

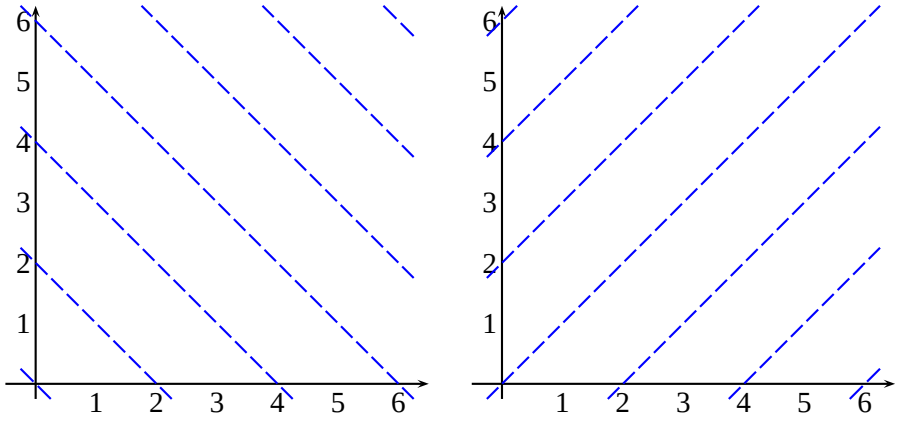
Proposition 5.9 *The operator '$\nabla_{\mathcal{G}}$' is a widening on $\mathbb{G}_n$.*

Proof. In order to show that '$\nabla_{\mathcal{G}}$' is a widening operator, we prove that conditions (1) and (2) in Definition 5.2 hold. Let $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$, $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2) \in \mathbb{G}_n$, where $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{G}_1$ is in minimal form and $\mathcal{G}_2$ is in strong minimal form.

By Definition 5.8, if $\mathcal{L}_1 = \varnothing$ or $\dim(\mathcal{L}_1) < \dim(\mathcal{L}_2)$, then $\mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2 = \mathcal{L}_2$. Therefore, in this case, condition (1) holds. Clearly, the empty grid can occur only as the first element of a strictly increasing chain of grids; moreover, if $\mathcal{L}$ and $\mathcal{L}'$ are any two successive and distinct grids in the increasing chain of condition (2) in Definition 5.2, then $0 \leq \dim(\mathcal{L}) \leq \dim(\mathcal{L}') \leq n$. Hence, the case when $\mathcal{L}_1 = \varnothing$ or $\dim(\mathcal{L}_1) < \dim(\mathcal{L}_2)$ hold can occur no more than a finite number of times in such a chain.

Suppose now that $\mathcal{L}_1 \neq \varnothing$ and $\dim(\mathcal{L}_1) = \dim(\mathcal{L}_2)$, so that the second case of the widening computation applies (note that, due to the inclusion hypothesis, $\dim(\mathcal{L}_1) > \dim(\mathcal{L}_2)$ cannot hold), and let $\mathcal{G}_S = (L_S, Q_S, P_S)$ where

$$P_S = P_2, \quad Q_S = \{\, \mathbf{v} \in Q_2 \mid \exists \mathbf{u} \in Q_1 \,.\, \mathbf{u} \Downarrow \mathbf{v} \,\}, \quad L_S = L_2 \cup (Q_2 \setminus Q_S).$$



(a) Grid $\mathcal{L}_1 \nabla_c \mathcal{L}_2$. (b) Grid $\mathcal{L}_1 \nabla_g \mathcal{L}_2$.

Figure 5.2: Comparing the two grid widenings.

Then, if $\mathrm{ggen}(\mathcal{G}_S) \subseteq \mathrm{ggen}(\mathcal{G}_2)$, we have that $L_S \subseteq L_2$, but by Definition 5.8, $L_2 \subseteq L_S$, therefore $L_2 = L_S$. Hence we must have that for all $\mathbf{q}_2 \in Q_2, \exists \mathbf{q}_1 \in Q_1$ such that $\mathbf{q}_1 \Downarrow \mathbf{q}_2$, so condition (1) holds. Now as Q_1 is pivot equivalent to Q_2, then $\#\, Q_1 = \#\, Q_2$. Let $\mathcal{C}_1 = (\mathcal{F}_1, \mathcal{E}_1)$ and $\mathcal{C}_2 = (\mathcal{F}_2, \mathcal{E}_2)$ be congruence systems for $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, as constructed in Lemma 3.28. Then by Lemma 3.34, we get that $\#\, \mathcal{E}_1 \geq \#\, \mathcal{E}_2$, but since $\dim(\mathcal{L}_1) = \dim(\mathcal{L}_2)$ and $\mathcal{L}_1, \mathcal{L}_2$ are both in minimal form we must have $\#\, \mathcal{E}_1 = \#\, \mathcal{E}_2$, hence by Lemma 3.28, $\#\, L_1 = \#\, L_2$. Now as $\mathcal{L}_1 \subseteq \mathcal{L}_2$ we must have that for all lines $\boldsymbol{\ell}_1 \in L_1$ such that $\mathrm{piv}_>(\boldsymbol{\ell}_1) = k$, there exists $\boldsymbol{\ell}_2 \in L_2$ such that $\mathrm{piv}_>(\boldsymbol{\ell}_2) = k$. Therefore $\mathcal{G}_1 \Downarrow \mathcal{G}_2$. Hence by Proposition 4.3, $\mathcal{L}_1 = \mathcal{L}_2$. Thus, if $\mathcal{L}_1 \neq \mathcal{L}_2$, we know $\#\, L_1 \leq \#\, L_2$ and $\mathrm{ggen}(\mathcal{G}_2) \subseteq \mathrm{ggen}(\mathcal{G}_S)$ so condition (1) holds and we have three cases to consider. The first is if $\#\, Q_1 < \#\, Q_2$, then $\#\, L_S \geq \#\, L_2 + \#\, Q_2 - \#\, Q_1 > \#\, L_1$. The second case is if $\#\, Q_1 = \#\, Q_2$, then there exists $\mathbf{u} \in Q_1$ with $\mathrm{piv}_>(\mathbf{u}) = k$ such that for all $\mathbf{v} \in Q_2$ if $\mathrm{piv}_>(\mathbf{v}) = k$, then $u_k \neq v_k$, hence $\mathbf{v} \in L_S$, therefore $\#\, L_S > \#\, L_1$. Finally for the last case suppose $\#\, Q_1 > \#\, Q_2$, then there exists $\mathbf{u} \in Q_1$ with $\mathrm{piv}_>(\mathbf{u}) = k$ such that for all $\mathbf{v} \in Q_2$, $\mathrm{piv}_>(\mathbf{v}) \neq k$, hence $\mathbf{v} \in L_S$, therefore $\#\, L_S > \#\, L_1$. Therefore condition (2) of Definition 5.2 holds. □

Assuming that $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$, $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$ and we know $\mathcal{L}_1 \subseteq \mathcal{L}_2$ this widening can be implemented to have complexity $\mathrm{O}(n^2)$, this is since all that is required is the copying of at most n generators from $\mathcal{L}_2$ to $\mathcal{L}_1 \nabla_g \mathcal{L}_2$. If however the generator system $\mathcal{G}_1$ is not in minimal form and the system $\mathcal{G}_2$ is not in strong minimal form, then the complexity of widening is that of the minimisations, namely $\mathrm{O}(mn \min\{m, n\})$, where $\#\, \mathcal{G}_1 = m_1, \#\, \mathcal{G}_2 = m_2$ and $m = \max\{m_1, m_2\}$. The following shows that the widenings for each representation are not always equivalent.

Example 5.10 *Consider grids* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1) = \mathrm{ggen}(\mathcal{G}_1)$ *and* $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{ggen}(\mathcal{G}_2)$

where

$$\mathcal{C}_1 = \{x \equiv_2 0,\ y \equiv_2 0\}, \qquad \mathcal{G}_1 = \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right),$$
$$\mathcal{C}_2 = \{x \equiv_1 0,\ x + y \equiv_2 0\}, \qquad \mathcal{G}_2 = \left(\varnothing, \begin{pmatrix} 1 & 0 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right).$$

Then it can be seen that $\mathcal{C}_2$ and $\mathcal{G}_2$ are in strong minimal form. $\mathcal{L}_1$ and $\mathcal{L}_2$ are the same as in Example 5.6 and can be seen in Figure 5.1(a) on Page 72. Then $\mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathrm{gcon}\big(\{x + y \equiv_2 0\}\big)$ can be seen in Figure 5.2(a) and

$$\mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2 = \left(\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right)$$

can be seen in Figure 5.2(b). Now by applying conversion to $\mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2$ we get $\mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2 = \mathrm{gcon}\big(\{x - y \equiv_2 0\}\big)$. Hence $\mathcal{L}_1 \nabla_c \mathcal{L}_2 \neq \mathcal{L}_1 \nabla_{\mathcal{G}} \mathcal{L}_2$.

Let us now consider a congruence widening for grids which is the natural counterpart to the standard widening for convex polyhedra as specified in the PhD thesis of N. Halbwachs [43], also described in [44]. It might be asked why we did not define our congruence widening in this way. To see why, consider the following grid extrapolation operator 'h'. Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) \in \mathbb{G}_n \setminus \varnothing$ where $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and $\mathcal{C}_1$ and $\mathcal{C}_2$ are congruence systems in $\mathbb{R}^n$ in strong minimal form. Then $h(\mathcal{L}_1, \mathcal{L}_2) := \mathrm{gcon}(\mathcal{C}_1' \cup \mathcal{C}_2')$ where

$$\mathcal{C}_1' := \Big\{ \beta \in \mathcal{C}_1 \;\Big|\; \mathcal{L}_2 \subseteq \mathrm{gcon}\big(\{\beta\}\big) \Big\}, \tag{5.1}$$
$$\mathcal{C}_2' := \Big\{ \gamma \in \mathcal{C}_2 \;\Big|\; \exists \beta \in \mathcal{C}_1 \,.\, \mathcal{L}_1 = \mathrm{gcon}\big(\mathcal{C}_1[\gamma/\beta]\big) \Big\}. \tag{5.2}$$

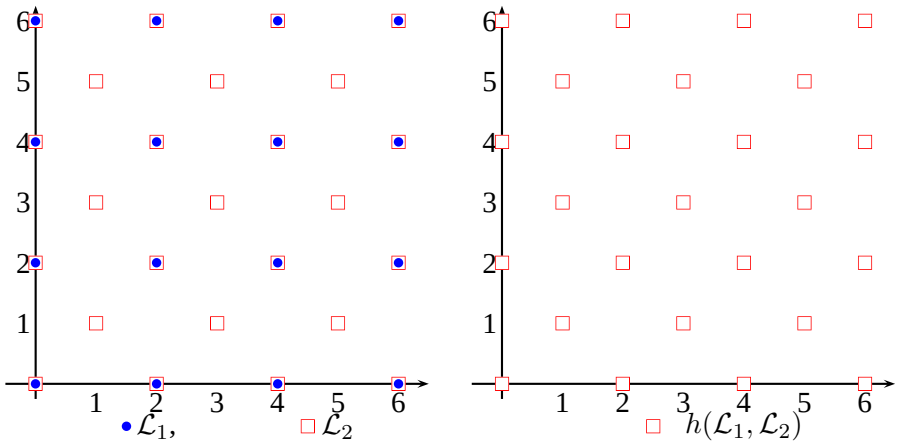
Example 5.11 will show that to ensure the operator 'h' is well-defined, *both* the congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ need to be in strong minimal form.

Example 5.11 *Let $\mathcal{L}_1 := \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 := \mathrm{gcon}(\mathcal{C}_2)$ where*

$$\mathcal{C}_1 = \{x \equiv_2 0,\ x + y \equiv_2 0\},$$
$$\mathcal{C}_2 = \{x \equiv_1 0,\ 3x + y \equiv_2 0\}.$$

Note that $\mathcal{L}_1$ and $\mathcal{L}_2$ are not in strong minimal form and can be seen in Figure 5.3(a). Then, assuming definition (5.1) for $\mathcal{C}_1'$ and (5.2) for $\mathcal{C}_2'$, we have $\mathcal{C}_1' = \{x + y \equiv_2 0\}$ and $\mathcal{C}_2' = \{3x + y \equiv_2 0\}$ so that

$$h(\mathcal{L}_1, \mathcal{L}_2) = \mathrm{gcon}\big(\mathcal{C}_1' \cup \mathcal{C}_2'\big) = \mathrm{gcon}\big(\{x + y \equiv_2 0, 3x + y \equiv_2 0\}\big)$$



(a) Grids $\mathcal{L}_1$ and $\mathcal{L}_2$. (b) Grid $h(\mathcal{L}_1, \mathcal{L}_2)$.

Figure 5.3: Grid Widening.

although applying the strong minimal form algorithm to $\mathcal{C}'_1 \cup \mathcal{C}'_2$ we get the equivalent congruence system

$$h(\mathcal{L}_1, \mathcal{L}_2) = \mathrm{gcon}\big(\{x \equiv_1 0, x + y \equiv_2 0\}\big).$$

The grid $h(\mathcal{L}_1, \mathcal{L}_2)$ can be seen in Figure 5.3(b).

Note also that, as there is no finite set of proper congruences semantically equivalent to a single equality, the above definition has ignored any distinction between equalities and proper congruences. Observe that the computation of the congruence system $\mathcal{C}'_2$ should be carefully done as a naive implementation will be expensive. A naive implementation, where it is assumed both grids are in strong minimal form, would have complexity $\mathrm{O}\big(n^5\big)$ as it would require testing for equality between $\mathcal{L}_1$ and each of the new grids where the n possible $\gamma \in \mathcal{C}_2$ replace each of the n possible $\beta \in \mathcal{C}_1$. Example 5.12 illustrates that ignoring the $\mathcal{C}'_2$ component (as can be done in implementations of the standard widening for convex polyhedra when the affine hull is the universe) can lose precision.

Example 5.12 *Consider again the grids and congruence systems in Example 5.6 on Page 72. Then, assuming definition (5.1) for $\mathcal{C}'_1$ and (5.2) for $\mathcal{C}'_2$ and $\mathcal{C}'_3$, we have $\mathcal{C}'_1 = \varnothing$, $\mathcal{C}'_2 = \{x + y \equiv_2 0\}$ and $\mathcal{C}'_3 = \{3x + y \equiv_2 0\}$ so that*

$$h(\mathcal{L}_1, \mathcal{L}_2) = \mathrm{gcon}\big(\mathcal{C}'_1 \cup \mathcal{C}'_2\big) = \mathrm{gcon}\big(\{x + y \equiv_2 0\}\big)$$

and

$$h(\mathcal{L}_1, \mathcal{L}_3) = \mathrm{gcon}\big(\mathcal{C}'_1 \cup \mathcal{C}'_3\big) = \mathrm{gcon}\big(\{3x + y \equiv_2 0\}\big).$$

Therefore, ignoring the $\mathcal{C}'_2$ component for $h(\mathcal{L}_1, \mathcal{L}_2)$ will result in the widening producing the set

$\mathbb{R}^2$ and ignoring the $\mathcal{C}'_3$ component for $h(\mathcal{L}_1, \mathcal{L}_3)$ will result in the widening producing the set $\mathbb{R}^2$.

5.3.1 Enhancements

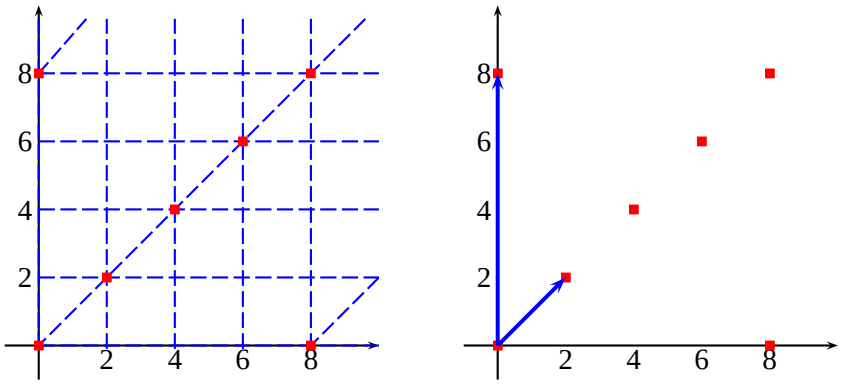
Often in analysis or verification, the convergence guarantee that comes with a widening operator is not essential and in such cases, all that is required are *extrapolation* operators. These differ from widenings in that their use along an upper iteration sequence does not ensure convergence in a finite number of steps. Therefore the widenings '∇_c' and '∇_g' can be used for developing more refined widenings or extrapolations by following techniques such as the frameworks proposed in [6,8] for convex polyhedra. The precision of a grid widening can also be improved by exploiting the covering boxes, defined in Section 4.6.1. That is, we can use boxes to provide a *covered* extrapolation operator that improves the approximation of the widening operator by ensuring that the result cannot be worse than the covering box for the larger of the two grids being widened.

One way to show that an extrapolation operator is, in fact, a widening is to provide the operator with a *finite convergence certificate* [8]. In particular, for the grid domain and widenings '∇_c' and '∇_g', such a certificate is defined to be a triple $(\mathcal{O}, \succ, \mu)$ where $(\mathcal{O}, \succ)$ is a well-founded ordered set and $\mu\colon \mathbb{G}_n \to \mathcal{O}$ is such that, for all $\mathcal{L}_1 \subset \mathcal{L}_2 \in \mathbb{G}_n$, $\mu(\mathcal{L}_1) \succ \mu(\mathcal{L}_3)$ where $\mathcal{L}_3 = \mathcal{L}_1 \nabla_c \mathcal{L}_2 = \mathcal{L}_1 \nabla_g \mathcal{L}_2$. Thus, a finite convergence certificate for both the grid widenings can be defined by taking $\mathcal{O}$ equal to $\{0, \ldots, n\} \times \{0, \ldots, n\}$, $\succ$ the lexicographic ordering on $\mathcal{O}$ and, for all $\mathcal{L} \in \mathbb{G}_n$, letting $\mu(\mathcal{L}) := (\#\,\mathcal{E}, \#\,\mathcal{C})$ where $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$, $\mathcal{C}$ is in minimal form, and $\mathcal{E} \subseteq \mathcal{C}$ is the set of equalities in $\mathcal{C}$. By Definitions 5.4 and 5.8 and Propositions 4.2 and 4.3, it follows that $\mathcal{L}_1 \neq \mathcal{L}_2$ implies $\mu(\mathcal{L}_1) \succ \mu(\mathcal{L}_3)$; hence we have the same finite convergence certificate for both the grid widenings. Observe that this implies that any iteration using a mixed sequence of congruence and generator grid widenings will converge after a finite number of steps.

It is shown in [8] that a widening for a powerset domain can be obtained from any widening on its base-level domain that has a finite convergence certificate. Thus, with the above certificate for the grid widenings, we can instantiate the generic widenings for powersets to one for powersets of grids, using any combination of the grid widenings '∇_c' and '∇_g'.

5.4 Weakly Relational Grid Domains

In [51, 53], Miné introduces a set of conditions to construct weakly relational domains. As noted in Section 3.8 one of the domains created was the domain of zone-congruences which requires that the congruences are defined equivalently to how the constraints of a bounded difference shape are defined. We now consider this domain and call it the bounded difference grid domain. Also we will specify a new weakly relational domain, called the octagonal grid domain, which has not been considered before and requires that the congruences are defined equivalently to how the constraints of an octagon are defined. Both of these domains are restricted versions of the grid domain that include the set of rectilinear grids.



(a) Grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$. (b) Grid $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$.

Figure 5.4: A bounded difference grid.

Definition 5.13 (Bounded Difference Congruences.) *Let* $\mathbf{a} \in \mathbb{R}^n$ *and* $f, b \in \mathbb{Q}$*, then the linear constraint* $\langle \mathbf{a}, \mathbf{v} \rangle \equiv_f b$ *is said to be a* bounded difference congruence *if and only if there exists two indices* $i, j \in \{1, \ldots, n\}$ *such that*

- $a_i, a_j \in \{-1, 0, 1\}$ *and* $a_i \neq a_j$
- $a_k = 0$*, for all* $k \notin \{i, j\}$.

Definition 5.14 (Bounded Difference Grid.) *A grid* $\mathcal{L}$ *is a* bounded difference grid *(BDG) if it can be* described by *a congruence system* $\mathcal{C}$ *in* $\mathbb{Q}^n$*, where* $\mathcal{C}$ *is a finite set of bounded difference congruences in* $\mathbb{Q}^n$*. That is* $\mathcal{L}$ *is a bdg if every vector of* $\mathcal{L}$ *satisfies all the congruences in* $\mathcal{C}$.

Note that a bounded difference grid is equivalent to an element of the zone-congruence domain. Recall form Definition 3.9 that a grid is rectilinear if it can be represented by a non-relational set of congruences therefore a bounded difference grid can be rectilinear. Example 5.4 gives the congruence and generator systems for a simple 2-dimensional bounded difference grid.

Example 5.15 *Let* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$ *where*

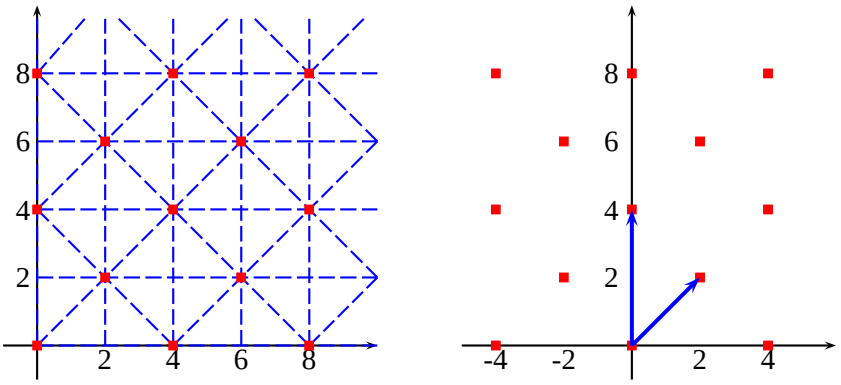
$$\mathcal{C} := \{x \equiv_2 0,\ y \equiv_2 0,\ -x + y \equiv_8 0\}.$$

Then $\mathcal{L}$ *is a bounded difference grid and is illustrated in Figure 5.4 by the points. The minimal form of* $\mathcal{C}$ *is* $\mathcal{C}'$ *where*

$$\mathcal{C}' := \{x \equiv_2 0,\ -x + y \equiv_8 0\}$$

and the generator system in minimal form is

$$\mathcal{G} := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 2 & 8 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right).$$



(a) Grid $\mathcal{L} = \text{gcon}(\mathcal{C})$. (b) Grid $\mathcal{L} = \text{ggen}(\mathcal{G})$.

Figure 5.5: An octagonal grid.

The congruences of $\mathcal{C}$ are illustrated by the dashed lines in Figure 5.4(a) and the parameters of $\mathcal{G}$ are illustrated by the arrows in Figure 5.4(b).

Definition 5.16 (Octagonal Congruences.) *Let $\mathbf{a} \in \mathbb{R}^n$ and $f, b \in \mathbb{Q}$, then the linear constraint $\langle \mathbf{a}, \mathbf{v} \rangle \equiv_f b$ is said to be an* octagonal congruence *if and only if there exists two indices $i, j \in \{1, \ldots, n\}$ such that*

- *$a_i, a_j \in \{-1, 0, 1\}$ and $a_i \neq 0$*
- *$a_k = 0$, for all $k \notin \{i, j\}$.*

Definition 5.17 (Octagonal Grid.) *A grid $\mathcal{L}$ is a* octagonal grid *(ogrid) if it can be* described by *a congruence system $\mathcal{C}$ in $\mathbb{Q}^n$, where $\mathcal{C}$ is a finite set of octagonal congruences in $\mathbb{Q}^n$. That is $\mathcal{L}$ is an ogrid if every vector of $\mathcal{L}$ satisfies all the congruences in $\mathcal{C}$.*

The set of octagonal grids is a subset of $\mathbb{G}_n$ that includes the set of rectilinear and bounded difference grids. Recall from Definition 3.9 that a grid is rectilinear if it can be represented by a non-relational set of congruences therefore an octagonal grid can be rectilinear. Example 5.5 gives the congruence and generator systems for a simple 2-dimensional ogrid.

Example 5.18 *Let $\mathcal{L} = \text{gcon}(\mathcal{C}) = \text{ggen}(\mathcal{G})$ where*

$$\mathcal{C} := \{x \equiv_2 0,\ y \equiv_2 0,\ x - y \equiv_4 0,\ x + y \equiv_4 0\}.$$

Then $\mathcal{L}$ is an octagonal grid and is illustrated in Figure 5.5 by the points. The minimal form of $\mathcal{C}$ is $\mathcal{C}'$ where

$$\mathcal{C}' := \{x \equiv_2 0,\ x + y \equiv_4 0\}$$

and the generator system in minimal form is

$$\mathcal{G} := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 2 & 4 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right).$$

The congruences of $\mathcal{C}$ are illustrated by the dashed lines in Figure 5.5(a) and the parameters of $\mathcal{G}$ are illustrated by the arrows in Figure 5.5(b).

Note that minimisation, conversion and the set of operations defined in Chapter 4 can be performed on the bounded difference and octagonal grids like they are in the case of a general grid that is not rectilinear.

5.5 Applications

In this section we discuss applications for the domain of rational grids. Many program properties are quantitative or depend on quantitative information and therefore have the potential to be approximated by the grid domain. While such information may depend directly on the values of numerical data objects, it could instead reflect some numerical measures of the structure of the program and its data. We first discuss applications where the values of numeric variables are abstracted.

Example 5.19 shows how the grid domain can be used to find non-trivial relational congruence properties not found using the polyhedra domain [32], constraint-based analysis [75] or polynomial invariants [74].

Example 5.19 *Consider again the program fragment from Example 3.2 on Page 24 which is now annotated with program points $\mathsf{P}j$, for $j = 1, \ldots, 5$:*

```
x := 2; y := 0;              (P1)
for i := 1 to m              (P2)
  if ... then
    x := x + 4               (P3)
  else
    x := x + 2; y := y + 1   (P4)
  endif                      (P5)
endfor
```

Let $\mathcal{L}_j^i \in \mathbb{G}_2$ denote the grid computed at the i-th iteration executed by the point $\mathsf{P}j$. Initially, $\mathcal{L}_j^0 = \varnothing = \mathrm{gcon}\big(\{1 = 0\}\big)$, for $j = 1, \ldots, 5$. After the first iteration of the loop we have the

following grids:

$$\begin{aligned}
\mathcal{L}_1^1 &= \mathrm{gcon}\big(\{x = 2,\ y = 0\}\big),\\
\mathcal{L}_2^1 &= \mathrm{gcon}\big(\{x = 2,\ y = 0\}\big),\\
\mathcal{L}_3^1 &= \mathrm{gcon}\big(\{x = 6,\ y = 0\}\big),\\
\mathcal{L}_4^1 &= \mathrm{gcon}\big(\{x = 4,\ y = 1\}\big),\\
\mathcal{L}_5^1 &= \mathrm{gcon}\big(\{x = 4,\ y = 1\}\big) \oplus \mathrm{gcon}\big(\{x = 6,\ y = 0\}\big)\\
&= \mathrm{gcon}\big(\{x + 2y = 6,\ x \equiv_2 0\}\big).
\end{aligned}$$

Then after the second iteration of the loop we have

$$\begin{aligned}
\mathcal{L}_2^2 &= \mathrm{gcon}\big(\{x = 2, y = 0\}\big) \oplus \mathrm{gcon}\big(\{x + 2y = 6, x \equiv_2 0\}\big)\\
&= \mathrm{gcon}\big(\{x + 2y \equiv_4 2, x \equiv_2 0\}\big).
\end{aligned}$$

Subsequent computation steps show that an invariant for `P2` *has already been computed since* $\mathcal{L}_3^2 = \mathcal{L}_3^1$, $\mathcal{L}_4^2 = \mathcal{L}_4^1$, $\mathcal{L}_5^2 = \mathcal{L}_5^1$ *so that* $\mathcal{L}_2^3 = \mathcal{L}_2^2$. *Thus at the end of the program, the congruences* $x + 2y \equiv_4 2$ *and* $x \equiv_2 0$ *hold. The grid described by these congruences is given in Figure 3.1 on Page 24.*

Observe that, using convex polyhedra, a similar analysis will find instead that the inequalities $x - 2y \geq 2$ *and* $y \geq 0$ *hold [32].*

Data dependence analysis for arrays —deciding if two elements of an array can refer to the same element and, if so, under what conditions— is required for advanced optimizing compilers as noted by Pugh in [70]. Granger showed in [37–39] that the domain of grids can be used for this analysis, the following example also shows this.

Example 5.20 *Consider the following program (adapted from a simple example given in [70]):*

```
for i := 0 to 100
  for j := 2i to 100
    A[i, 2j + 1] := A[i, 2j]
  endfor
endfor
```

Then, the program reads from array elements $(0, 0)$, $(0, 2)$, …, $(0, 200)$, $(1, 4)$, … *and writes to array elements* $(0, 1)$, $(0, 3)$, …, $(0, 201)$, $(1, 5)$, …. *The two sets of points generate, respectively, the two grids* $\mathcal{L}_r$ *and* $\mathcal{L}_w$ *in* $\mathbb{R}^2$: $\mathcal{L}_r = \mathrm{ggen}\big\{(0, 0), (0, 2), (1, 4)\big\}$ *includes all the array elements that are read from, while* $\mathcal{L}_w = \mathrm{ggen}\big\{(0, 1), (0, 3), (1, 5)\big\}$ *includes all the array elements that are written to. Figure 5.6 illustrates the grids* $\mathcal{L}_r$ *and* $\mathcal{L}_w$ *where squares denote the points of the*

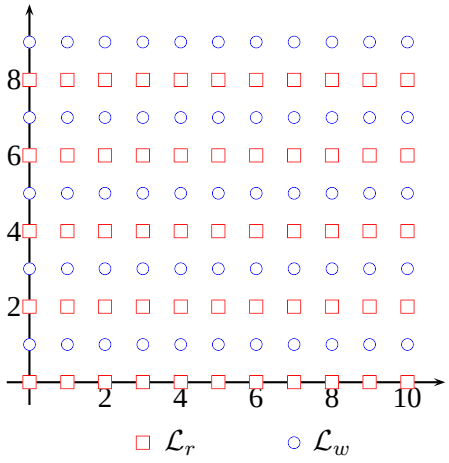


Figure 5.6: Example 5.20.

grid $\mathcal{L}_r$ and circles denote the points of the grid $\mathcal{L}_w$. Then it can be seen that the intersection $\mathcal{L}_r \cap \mathcal{L}_w$ is empty and that no location is both read and written.

As noted in Section 3.8 the domain of grids can also be used to estimate the worst case execution time of a program given a specific system [19], to aid in the construction of a comprehensive set of program transformations for saving energy on low-power architectures and improving performance on multimedia processors [47] and to gather information about non-linear operations within the program [46].

5.6 Related Work

As Granger's early work [39] and the $\mathbb{Z}$-polyhedra papers [68,71,72] only consider integer grids, a widening is not required as integer grids satisfy the ascending chain condition. In the Muller-Olm and Seidl paper [61], although congruences over rationals are considered a widening is not given. In [41, Proposition 10], Granger gives a widening for non-relational rational grids that returns a line parallel to an axis whenever the modulus for that dimension changes. It is then proposed that a generalized form of this could be used as a widening for all rational grids; however, exactly how this is to be done is not given in this paper. In Granger's thesis [38, Page 159], it is proposed that for all rational grids a parameter from the representation of one of the grids (or possibly some other vector for example one parallel to an axis) is chosen to be the enlargement vector $\mathbf{e}$.[1] Let $\mathcal{L}_i = \text{ggen}\big(\varnothing, Q_i, \{\mathbf{p}\}\big)$ and $Q_i = \{\mathbf{q}_{i1}, \ldots, \mathbf{q}_{in}\}$ for $i = 1, 2$. Then without loss of generality suppose $\mathbf{e} = \mathbf{q}_{21}$. For each of the grids a value μ is then calculated such that

$$\mu_i := \min\{\lambda \in \mathbb{Q}^+ | \lambda \mathbf{e} = \lambda_1 \mathbf{q}_{i1} + \ldots + \lambda_n \mathbf{q}_{in}\}.$$

[1] for a definition of this property see [38, Page 160]

If $\mu_1 > \mu_2$ then

$$\mathcal{L}_1 \nabla \mathcal{L}_2 := \mathrm{ggen}\big(\{\mathbf{e}\}, Q_2 \setminus \{\mathbf{e}\}, \{\mathbf{p}\}\big).$$

Example 5.21 illustrates this.

Example 5.21 *Let $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$, where*

$$\mathcal{G}_1 := \left(\varnothing, \begin{pmatrix} 4 & 0 \\ 2 & 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right) \text{ and } \mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right).$$

Then letting $\mathbf{e} = (2, 1)$ in $\mathcal{G}_2'$ we get that $\mu_1 = 2$ and $\mu_2 = 1$. This will produce the widening

$$\mathcal{L}_1 \nabla \mathcal{L}_2 := \mathrm{ggen}\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right).$$

Although in Example 5.21 both grids are represented in minimal form it is not actually a requirement of the widening. Granger states that there are possibly infinite different widenings depending on the choice of the enlargement vector $\mathbf{e}$. The following example shows that Granger's proposed widening is not actually a widening due to there being no restriction on the choice of the enlargement vector.

Example 5.22 *Let $\mathcal{L}_1 = \mathrm{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathrm{ggen}(\mathcal{G}_2)$, where*

$$\mathcal{G}_1 := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 1 & 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right) \text{ and } \mathcal{G}_2 := \left(\varnothing, \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right).$$

Then letting $\mathbf{e} = (2, 1)$ we get that $\mu_1 = 1$ and $\mu_2 = 1$. Hence the widening does not perform any enlargement. It can also be seen from this example that the fact that the representations for the grids are in minimal form makes no difference to the result.

In [51, 53], Miné introduces a basis from which to construct *weakly relational domains.* As noted in Section 3.8, the zone-congruence domain considers congruences which have the form $x - y = b \pmod f$ where b and f are rationals and are represented by a constraint matrix. Miné gives operations for intersection and join which are equivalent to the ones described here and also a widening which is equivalent to the one described by Granger in [41].

5.7 Conclusion

We have defined two widenings, one that uses the congruence representation and one that uses the generator representation. We have shown that the widenings come with simple syntactic checks and have efficient implementations. The widenings have complexity $\mathrm{O}(n^2)$ if the first grid is known to have its representation in minimal form and the second grid is known to have its representation in strong minimal form, otherwise the worst case complexity is that of the

minimisation algorithms. Unfortunately as yet we have not found an actual real life application that will require the grid widening. We have also defined two weakly relational grid domains, the bounded difference grid and the octagonal grid.

Chapter 6

The Grid-Polyhedron Domain

6.1 Introduction

In this chapter we consider the products of n-dimensional geometric domains and, in particular, the product of a grid with a polyhedron domain. Section 6.2 introduces the generic product of domains represented by sets of points in $\mathbb{R}^n$ and Section 6.3 introduces the partially reduced product which allows a range of interaction between the component domains. In Section 6.4 we introduce the techniques for ensuring the bounding hyperplanes of the polyhedron component contain at least one point in the grid component and in Section 6.5 we give the main abstract operations and the methods for their computation.

6.2 The Product Domain

Definition 6.1 (Product Domain.) *Let $A_1, A_2 \subseteq \wp\big(\wp(\mathbb{R}^n)\big)$ be two n-dimensional geometric domains. Then the* product $A_1 \times A_2$ *(also denoted by (A_1, A_2)) is the set $A \subseteq \wp\big(\wp(\mathbb{R}^n)\big)$ where*

$$A := \{a_1 \cap a_2 | a_1 \in A_1, a_2 \in A_2\}.$$

Definition 6.2 (Grid-Polyhedron.) *Let $\mathcal{P}$ be a polyhedron in $\mathbb{CP}_n$ and $\mathcal{L}$ a grid in $\mathbb{G}_n$. Then we say that $\mathcal{H} = (\mathcal{L}, \mathcal{P}) := \mathcal{L} \cap \mathcal{P}$ is a* grid-polyhedron. *The* grid-polyhedron domain $\mathbb{GP}_n$ *is the set of all grid-polyhedra in $\mathbb{R}^n$ ordered by the set inclusion relation, so that $\varnothing$ and $\mathbb{R}^n$ are the bottom and top elements of $\mathbb{GP}_n$ respectively.*

Example 6.3 *Let us consider a small example to show how the grid-polyhedron domain can be used to interpret a simple piece of code. Figure 1.3 on Page 6 illustrates the grid by the square*

points and the polyhedron by the shaded area. Therefore the grid-polyhedron is the set of grid points that lie within the bounded shaded area. Consider first the following program fragment for any value of `m`*:*

```
x := 0; y := 1                (P1)
for i := 1 to m               (P2)
  if ... then
    x := x + 3                (P3)
  else
    x := x + 2; y := y + 1    (P4)
  endif                       (P5)
endfor
```

Let $\mathcal{L}_j^i \in \mathbb{G}_2$ *denote the grid computed at the* i*-th iteration executed by the point* `P`j*. Initially,* $\mathcal{L}_j^0 = \varnothing = \mathrm{gcon}\big(\{1 = 0\}\big)$*, for* $j = 1, \ldots, 5$*. After the first iteration of the loop we have the following grids:*

$$\begin{aligned}
\mathcal{L}_1^1 &= \mathrm{gcon}\big(\{x = 0,\ y = 1\}\big),\\
\mathcal{L}_2^1 &= \mathrm{gcon}\big(\{x = 0,\ y = 1\}\big),\\
\mathcal{L}_3^1 &= \mathrm{gcon}\big(\{x = 3,\ y = 1\}\big),\\
\mathcal{L}_4^1 &= \mathrm{gcon}\big(\{x = 2,\ y = 2\}\big),\\
\mathcal{L}_5^1 &= \mathrm{gcon}\big(\{x = 3,\ y = 1\}\big) \oplus \mathrm{gcon}\big(\{x = 2,\ y = 2\}\big)\\
&= \mathrm{gcon}\big(\{x + y = 4,\ x \equiv_1 0\}\big).
\end{aligned}$$

Then after the second iteration of the loop we have

$$\begin{aligned}
\mathcal{L}_2^2 &= \mathrm{gcon}\big(\{x = 0, y = 1\}\big) \oplus \mathrm{gcon}\big(\{x + y = 4, x \equiv_1 0\}\big)\\
&= \mathrm{gcon}\big(\{x + y \equiv_3 1, x \equiv_1 0\}\big).
\end{aligned}$$

Subsequent computation steps show that an invariant for `P2` *has already been computed since* $\mathcal{L}_3^2 = \mathcal{L}_3^1$*,* $\mathcal{L}_4^2 = \mathcal{L}_4^1$*,* $\mathcal{L}_5^2 = \mathcal{L}_5^1$ *so that* $\mathcal{L}_2^3 = \mathcal{L}_2^2$*. Thus at the end of the program, the grid is given by* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ *where* $\mathcal{C}_\mathcal{L} := \{x \equiv_1 0, x + y \equiv_3 1\}$*. Observe that the grid* $\mathcal{L}$ *is also given by the generator system* $\mathcal{G}_\mathcal{L}$ *where*

$$\mathcal{G}_\mathcal{L} := \left(\varnothing, \begin{pmatrix} 1 & 0 \\ -1 & 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right).$$

Now consider the program fragment assuming that the value of `m = 4`*. Then let* $\mathcal{P}_j^i$ *denote the polyhedron computed in the* i*-th iteration at point* `P`j*. Initially* $\mathcal{P}_j^0 = \mathrm{con}\big(\{x = 0, y = 1\}\big)$*, for*

$j = 1, \ldots, 5$. *Then after the first iteration of the for loop we have the following polyhedra:*

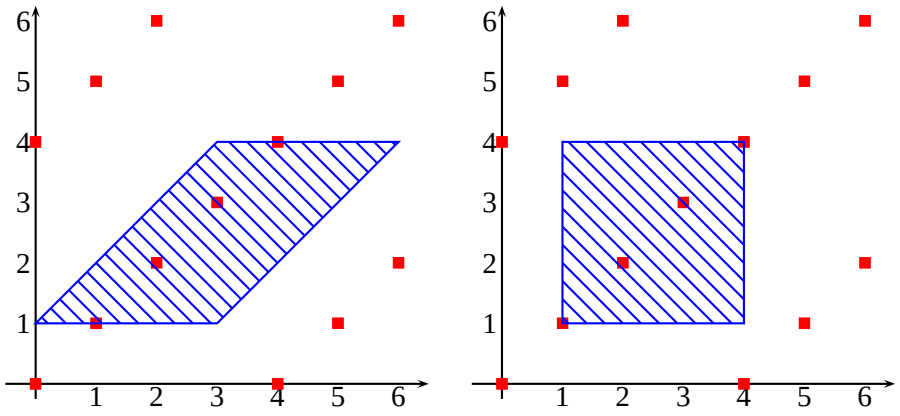
$$\begin{aligned}
\mathcal{P}_1^1 &= \mathrm{con}\big(\{x = 0, y = 1\}\big),\\
\mathcal{P}_2^1 &= \mathrm{con}\big(\{x = 0, y = 1\}\big),\\
\mathcal{P}_3^1 &= \mathrm{con}\big(\{x = 3, y = 1\}\big),\\
\mathcal{P}_4^1 &= \mathrm{con}\big(\{x = 2, y = 2\}\big),\\
\mathcal{P}_5^1 &= \mathrm{con}\big(\{x = 3, y = 1\}\big) \oplus \mathrm{con}\big(\{x = 2, y = 2\}\big)\\
&= \mathrm{con}\big(\{2 \le x \le 3, x + y \le 4\}\big).
\end{aligned}$$

Then after the second iteration of the loop we have the polyhedra:

$$\begin{aligned}
\mathcal{P}_2^2 &= \mathrm{con}\big(\{x = 0, y = 1\}\big) \oplus \mathrm{con}\big(\{2 \le x \le 3, x + y \le 4\}\big)\\
&= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 4\}\big),\\
\mathcal{P}_3^2 &= \mathrm{con}\big(\{1 \le y, 1 \le x - 2y, x + y \le 7\}\big),\\
\mathcal{P}_4^2 &= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 7\}\big),\\
\mathcal{P}_5^2 &= \mathrm{con}\big(\{1 \le y, 1 \le x - 2y, x + y \le 7\}\big) \oplus \mathrm{con}\big(\{2 \le y, -2 \le x - 2y, x + y \le 7\}\big)\\
&= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, 4 \le x + y \le 7\}\big),
\end{aligned}$$

and after the third and fourth iterations of the loop we have the polyhedra:

$$\begin{aligned}
\mathcal{P}_2^3 &= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 4\}\big) \oplus \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, 4 \le x + y \le 7\}\big)\\
&= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 7\}\big),\\
\mathcal{P}_3^3 &= \mathrm{con}\big(\{1 \le y, 1 \le x - 2y, x + y \le 10\}\big),\\
\mathcal{P}_4^3 &= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 10\}\big),\\
\mathcal{P}_5^3 &= \mathrm{con}\big(\{1 \le y, 1 \le x - 2y, x + y \le 10\}\big) \oplus \mathrm{con}\big(\{2 \le y, -2 \le x - 2y, x + y \le 10\}\big)\\
&= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, 4 \le x + y \le 10\}\big),\\
\mathcal{P}_2^4 &= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 7\}\big) \oplus \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, 4 \le x + y \le 10\}\big)\\
&= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 10\}\big),\\
\mathcal{P}_3^4 &= \mathrm{con}\big(\{1 \le y, 1 \le x - 2y, x + y \le 13\}\big),\\
\mathcal{P}_4^4 &= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, x + y \le 13\}\big),\\
\mathcal{P}_5^4 &= \mathrm{con}\big(\{1 \le y, 1 \le x - 2y, x + y \le 13\}\big) \oplus \mathrm{con}\big(\{2 \le y, -2 \le x - 2y, x + y \le 13\}\big)\\
&= \mathrm{con}\big(\{1 \le y, -2 \le x - 2y, 4 \le x + y \le 13\}\big).
\end{aligned}$$



(a) Grid-polyhedron $\mathcal{H}_1 = (\mathcal{L}, \mathcal{P}_1)$. (b) Grid-polyhedron $\mathcal{H}_2 = (\mathcal{L}, \mathcal{P}_2)$.

Figure 6.1: Equivalent grid-polyhedra.

Therefore the final polyhedron $\mathcal{P}$ at the end of four iterations has a constraint system given by

$$\begin{aligned}\mathcal{C}_{\mathcal{P}} &= \{1 \le y, -2 \le x - 2y, x + y \le 10\} \oplus \{1 \le y, -2 \le x - 2y, 4 \le x + y \le 13\} \\ &= \{1 \le y, -2 \le x - 2y, x + y \le 13\}.\end{aligned}$$

The polyhedron can also be defined by the vertices given by $\binom{0}{1}, \binom{8}{5}$ *and* $\binom{12}{1}$. *The polyhedron $\mathcal{P}$ can be seen in Figure 1.3 on Page 6. The grid-polyhedron is the set of points inside the polyhedron given by* $\binom{0}{1}, \binom{3}{1}, \binom{2}{2}, \binom{6}{1}, \binom{5}{2}, \binom{4}{3}, \binom{9}{1}, \binom{8}{2}, \binom{7}{3}, \binom{6}{4}, \binom{12}{1}, \binom{11}{2}, \binom{10}{3}, \binom{9}{4}$ *and* $\binom{8}{5}$. *It can be seen in Figure 1.3 that all the polyhedron constraints intersect grid-polyhedron points and that the polyhedron is reduced with respect to the grid points. Note that if we had considered the polyhedron for the program fragment for any value of* `m` *the constraint system would be given by* $\mathcal{C}_{\mathcal{P}}{}' = \{1 \le y, -2 \le x - 2y\}$.

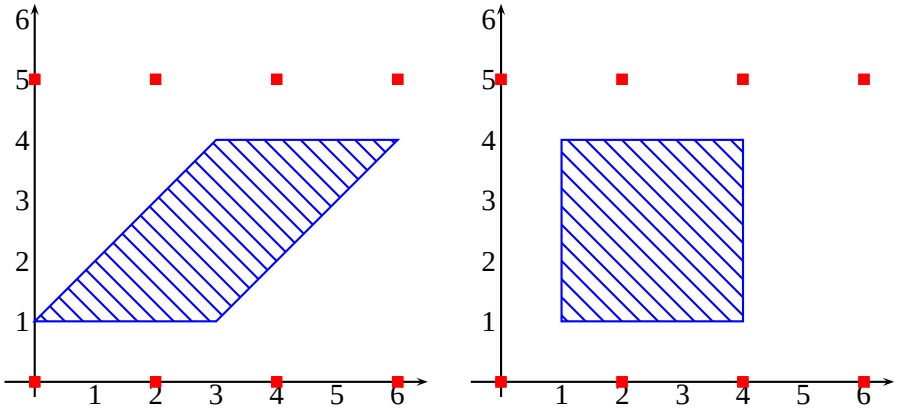
Although this section is considering the grid-polyhedron domain, many of the methods and algorithms suggested will be applicable to the more restricted polyhedra domains such as the Intervals [56], BDS [49] and Octagons [52], which will be considered in Chapter 7. We will also discuss later how each operation is effected depending on the type of domain we choose to put with the grids. First we observe that as elements of the grid-polyhedron domain denote the intersection of their components, the elements of the domain do not have a canonical form.

Let $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ be grid-polyhedra. Then as $\mathcal{H}_1$ is defined as the intersection of $\mathcal{L}_1$ and $\mathcal{P}_1$ and $\mathcal{H}_2$ is defined as the intersection of $\mathcal{L}_2$ and $\mathcal{P}_2$ it is possible to have $\mathcal{H}_1 = \mathcal{H}_2$ where $\mathcal{L}_1 = \mathcal{L}_2$ but $\mathcal{P}_1 \neq \mathcal{P}_2$. Example 6.4 illustrates this.

Example 6.4 *Consider the grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}, \mathcal{P}_2)$ in $\mathbb{GP}_2$ where*

$$\mathcal{L} := \mathrm{gcon}\big(\{x \equiv_1 0,\ -x + y \equiv_4 0\}\big),$$

$$\mathcal{P}_1 := \mathrm{con}\big(\{1 \le x \le 6,\ -2 \le -x + y \le 1\}\big) \quad \textit{and} \quad \mathcal{P}_2 := \mathrm{con}\big(\{1 \le x \le 4,\ 1 \le y \le 4\}\big).$$



(a) Grid-polyhedron $\mathcal{H}_1 = (\mathcal{L}, \mathcal{P}_1)$. (b) Grid-polyhedron $\mathcal{H}_2 = (\mathcal{L}, \mathcal{P}_2)$.

Figure 6.2: Equivalent grid-polyhedra that are empty.

The grid $\mathcal{L}$ is illustrated by the filled squares and the polyhedron $\mathcal{P}_1$ is illustrated by the bounded region in Figure 6.1(a). The grid $\mathcal{L}$ is illustrated by the filled squares and the polyhedron $\mathcal{P}_2$ is illustrated by the bounded region in Figure 6.1(b). Then it can be seen from Figure 6.1 that $\mathcal{H}_1$ and $\mathcal{H}_2$ are equivalent.

Example 6.5 shows that the representation of an empty grid-polyhedron is not canonical and that an empty grid-polyhedron can be defined using a non-empty grid and polyhedron.

Example 6.5 *Consider the grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}, \mathcal{P}_1)$ in $\mathbb{GP}_2$ where*

$$\mathcal{L} := \mathrm{gcon}\big(\{x \equiv_2 0,\ y \equiv_5 0\}\big),$$

$$\mathcal{P}_1 := \mathrm{con}\big(\{1 \leq x \leq 6,\ -2 \leq -x + y \leq 1\}\big) \quad \textit{and} \quad \mathcal{P}_2 := \mathrm{con}\big(\{1 \leq x \leq 4,\ 1 \leq y \leq 4\}\big).$$

The grid $\mathcal{L}$ is illustrated by the filled squares and the polyhedron $\mathcal{P}_1$ is illustrated by the bounded region in Figure 6.2(a). The grid $\mathcal{L}$ is illustrated by the filled squares and the polyhedron $\mathcal{P}_2$ is illustrated by the bounded region in Figure 6.2(b). Then it can be seen from Figure 6.2 that $\mathcal{H}_1$ and $\mathcal{H}_2$ are equivalent and both are empty.

It follows that it is desirable that elements of the domain have their components minimised, which, in the case of the grid-polyhedron domain, would make every element canonical. However the full minimisation operation for a grid-polyhedron has a high complexity cost (as it will involve the simplex method [67, 76], see Section 6.6.2) and also has the potential to adversely affect the widening operation and actually turns a widening on the component into no more than an extrapolation operation on the product with no fix-point guaranteed (see Section 6.6). Thus to provide a framework for a choice of interaction, we introduce here the Partially Reduced Product.

6.3 The Partially Reduced Product

The partially reduced product is defined so as to allow a range of interaction between the component domains. To achieve this, it is provided with additional reduction operations as parameters.

Definition 6.6 (Partially Reduced Product Domain.) *Let $A_1, A_2 \subseteq \wp\big(\wp(\mathbb{R}^n)\big)$ be two n-dimensional geometric domains and $\sigma^1 : A_1 \times A_2 \to A_1$ and $\sigma^2 : A_1 \times A_2 \to A_2$ be two operations on them such that*

$$\sigma^1(a_1, a_2) \subseteq a_1 \quad \textit{and} \quad \sigma^1(a_1, a_2) \cap a_2 = a_1 \cap a_2,$$
$$\sigma^2(a_1, a_2) \subseteq a_2 \quad \textit{and} \quad a_1 \cap \sigma^2(a_1, a_2) = a_1 \cap a_2.$$

Then the triple $(A_1 \times A_2, \sigma^1, \sigma^2)$ is a partially reduced product domain.

Illustration 6.7 *Consider the product domain $A \subseteq \wp\big(\wp(\mathbb{R}^n)\big)$, where $A_1 \times A_2$ represents A, and the operations σ^1_D and σ^2_D are defined so that $\sigma^1_D(a_1, a_2) = a_1$ and $\sigma^2_D(a_1, a_2) = a_2$. Then the triple $(A_1 \times A_2, \sigma^1_D, \sigma^2_D)$ is a partially reduced product domain which we call the* direct *product domain [28].*

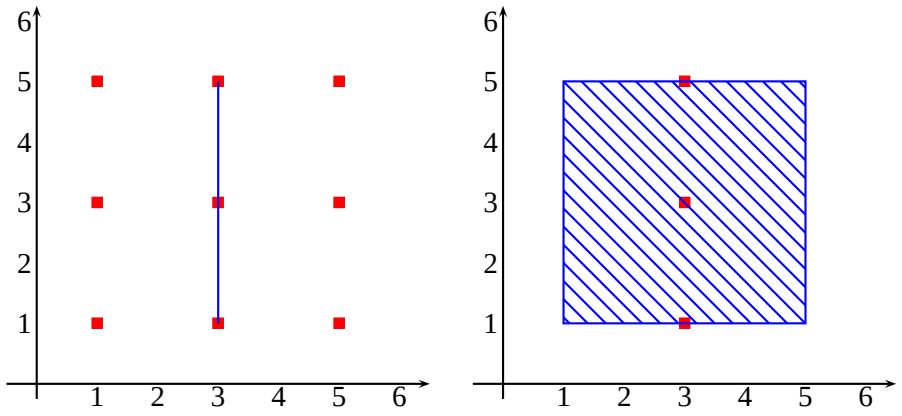
Illustration 6.8 *Consider the product domain $A \subseteq \wp\big(\wp(\mathbb{R}^n)\big)$, where $A_1 \times A_2$ represents A, and the operations σ^1_R and σ^2_R are defined so that $\sigma^1_R(a_1, a_2) = a'_1$, where $a'_1 \in A_1$ is the minimal element such that $a_1 \cap a_2 = a'_1 \cap a_2$, and $\sigma^2_R(a_1, a_2) = a'_2$, where $a'_2 \in A_2$ is the minimal element such that $a_1 \cap a_2 = a_1 \cap a'_2$. Then the triple $(A_1 \times A_2, \sigma^1_R, \sigma^2_R)$ is a partially reduced product domain which we call the* reduced *product domain [28].*

Illustration 6.9 *Consider the product domain $A \subseteq \wp\big(\wp(\mathbb{R}^n)\big)$, where $A_1 \times A_2$ represents A, and the operations $\sigma^1_\varnothing$ and $\sigma^2_\varnothing$ are defined so that $\sigma^1_\varnothing(a_1, a_2) = \sigma^2_\varnothing(a_1, a_2) = \varnothing$ if either $a_1 = \varnothing$ or $a_2 = \varnothing$ and $\sigma^1_\varnothing(a_1, a_2) = A_1$ and $\sigma^2_\varnothing(a_1, a_2) = A_2$, otherwise. Then the triple $(A_1 \times A_2, \sigma^1_\varnothing, \sigma^2_\varnothing)$ is a partially reduced product domain which we call the* smash *product domain.*

Let us now consider the partially reduced product where the component domains are those of the grids and polyhedra. Then we can specialise each of these definitions to the grid-polyhedron domain.

Illustration 6.10 *Consider the operations σ^1_D and σ^2_D defined in Illustration 6.7 together with the grid-polyhedron domain $\mathbb{GP}_n$. Then the triple $(\mathbb{GP}_n, \sigma^1_D, \sigma^2_D)$ is a partially reduced grid-polyhedron domain which we call the* direct *product domain [28].*

Illustration 6.11 *Consider the operations σ^1_R and σ^2_R defined in Illustration 6.8 together with the grid-polyhedron domain $\mathbb{GP}_n$. Then the triple $(\mathbb{GP}_n, \sigma^1_R, \sigma^2_R)$ is a partially reduced grid-polyhedron domain which we call the* reduced *product domain [28]. If $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ and $\sigma^1_R(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma^2_R(\mathcal{L}, \mathcal{P}) = \mathcal{P}$, then we say that the pair $(\mathcal{L}, \mathcal{P})$ is a* reduced product.



(a) Grid-polyhedron $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$. (b) Grid-polyhedron $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$.

Figure 6.3: Examples where equalities could be shared.

Illustration 6.12 *Consider the operations $\sigma^1_{\varnothing}$ and $\sigma^2_{\varnothing}$ defined in Illustration 6.9 together with the grid-polyhedron domain $\mathbb{GP}_n$. Then the triple $(\mathbb{GP}_n, \sigma^1_{\varnothing}, \sigma^2_{\varnothing})$ is a partially reduced grid-polyhedron domain which we call the* smash *product domain. If $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ and $\sigma^1_{\varnothing}(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma^2_{\varnothing}(\mathcal{L}, \mathcal{P}) = \mathcal{P}$, then we say that the pair $(\mathcal{L}, \mathcal{P})$ is a* smash product.

As the affine space of a grid-polyhedron is the intersection of the affine spaces of the component domains, any equalities in the constraint or congruence representation of one component can be added to the representation of the other component. So, as a grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ and $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$, is defined as the intersection of $\mathcal{L}$ and $\mathcal{P}$, each point in $\mathcal{H}$ must satisfy all the congruences in $\mathcal{C}_{\mathcal{L}}$ and all the constraints in $\mathcal{C}_{\mathcal{P}}$; moreover, equalities can be represented both as congruences and as constraints. Thus sharing equality information between the components of the product is safe and can minimise the component domains possibly leading to a detection of emptiness. Example 6.13 illustrates how equality information can be shared.

Example 6.13 *Consider the grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$ where*

$$\mathcal{L}_1 := \mathrm{gcon}\big(\{x \equiv_2 1,\ y \equiv_2 1\}\big) \quad \textit{and} \quad \mathcal{L}_2 := \mathrm{gcon}\big(\{x = 3,\ y \equiv_2 1\}\big),$$
$$\mathcal{P}_1 := \mathrm{con}\big(\{x = 3,\ 1 \leq y \leq 5\}\big) \quad \textit{and} \quad \mathcal{P}_2 := \mathrm{con}\big(\{1 \leq x \leq 5,\ 1 \leq y \leq 5\}\big).$$

The grid-polyhedron $\mathcal{H}_1$ is illustrated by the filled squares on the line in Figure 6.3(a) and the grid-polyhedron $\mathcal{H}_2$ is illustrated by the filled squares in Figure 6.3(b). Therefore it can be seen from Figure 6.3(a) and Figure 6.3(b) that $\mathcal{H}_1 = \mathcal{H}_2 = (\mathcal{L}, \mathcal{P})$, where

$$\mathcal{L} := \mathrm{gcon}\big(\{x = 3,\ y \equiv_2 1\}\big) \quad \textit{and} \quad \mathcal{P} := \mathrm{con}\big(\{x = 3,\ 1 \leq y \leq 5\}\big).$$

Therefore we can specialise the partially reduced product again for the grid-polyhedron domain.

Illustration 6.14 *Consider again the partially reduced product domain* $(\mathbb{GP}_n, \sigma^1_=, \sigma^2_=)$ *where, for all* $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$*,*

$$\mathrm{affine.hull}\big(\sigma^1_=(\mathcal{L}, \mathcal{P})\big) = \mathrm{affine.hull}\big(\sigma^2_=(\mathcal{L}, \mathcal{P})\big) = \mathrm{affine.hull}\big(\mathcal{H}\big).$$

Then the triple $(\mathbb{GP}_n, \sigma^1_=, \sigma^2_=)$ *is a partially reduced grid-polyhedron domain which we call the* constraint *product domain. If* $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ *and* $\sigma^1_=(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ *and* $\sigma^2_=(\mathcal{L}, \mathcal{P}) = \mathcal{P}$*, then we say that the pair* $(\mathcal{L}, \mathcal{P})$ *is a* constraint product*.*

Given any grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$*, it is straightforward to compute a constraint product* $(\mathcal{L}', \mathcal{P}')$ *such that* $\mathcal{H} = (\mathcal{L}, \mathcal{P}) = (\mathcal{L}', \mathcal{P}')$*. To see this, suppose that* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ *and* $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$*, where* $\mathcal{C}_\mathcal{L}$ *and* $\mathcal{C}_\mathcal{P}$ *are in minimal form, then*

$$\mathcal{L}' := \sigma^1_=(\mathcal{L}, \mathcal{P}) = \mathrm{gcon}\Big(\mathcal{C}_\mathcal{L} \cup \big\{\langle \mathbf{v}, \mathbf{x}\rangle \equiv_0 d \;\big|\; \langle \mathbf{v}, \mathbf{x}\rangle = d \in \mathcal{C}_\mathcal{P}\big\}\Big)$$
$$\mathcal{P}' := \sigma^2_=(\mathcal{L}, \mathcal{P}) = \mathrm{con}\Big(\mathcal{C}_\mathcal{P} \cup \big\{\langle \mathbf{v}, \mathbf{x}\rangle = d \;\big|\; \langle \mathbf{v}, \mathbf{x}\rangle \equiv_0 d \in \mathcal{C}_\mathcal{L}\big\}\Big).$$

6.4 Tight and Weakly Tight Products

The partially reduced products defined so far only allow a very limited interaction between the components. We discuss now how a domain such as the grid-polyhedron constraints product can be further specialised by ensuring that the bounding hyperplanes of the polyhedron component contain at least one point in the grid component or even a point of the product itself. Let us now specialise the partially reduced product again for the grid-polyhedron domain and define what it is for a grid-polyhedron to be a tight or weakly tight product.

Definition 6.15 (Weakly Tight Product.) *Let the triple* $(\mathbb{GP}_n, \sigma^1_W, \sigma^2_W)$ *be a partially reduced grid-polyhedron domain where, for all* $\mathcal{H} \in \mathbb{GP}_n$*, there exists* $\mathcal{L}' \in \mathbb{G}_n$ *and* $\mathcal{P}' \in \mathbb{CP}_n$ *such that*

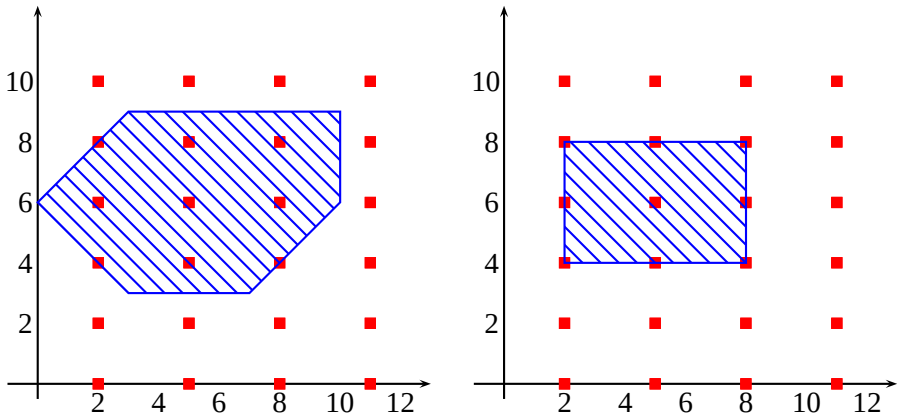
$$\mathcal{L}' := \sigma^1_W(\mathcal{L}, \mathcal{P}) = \sigma^1_=(\mathcal{L}, \mathcal{P}),$$
$$\mathcal{P}' := \sigma^2_W(\mathcal{L}, \mathcal{P}) = \sigma^2_=(\mathcal{L}, \mathcal{P})$$

and for some constraint system $\mathcal{C}_\mathcal{P}{}'$ *for* $\mathcal{P}'$*, for all* $\big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big) \in \mathcal{C}_\mathcal{P}{}'$*, there exists a point* $\mathbf{w} \in \mathcal{L}'$ *such that* $\langle \mathbf{v}, \mathbf{w}\rangle = d$*. Then* $(\mathbb{GP}_n, \sigma^1_W, \sigma^2_W)$ *is a* weakly tight *product domain and we say that* $\mathcal{C}_\mathcal{P}{}'$ *is a weakly tight polyhedron constraint system for* $\mathcal{H}$*.*

If $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ *and* $\sigma^1_W(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ *and* $\sigma^2_W(\mathcal{L}, \mathcal{P}) = \mathcal{P}$*, then we say that the pair* $(\mathcal{L}, \mathcal{P})$ *is a* weakly tight product*.*

Definition 6.16 (Tight Product.) *Let* $(\mathbb{GP}_n, \sigma^1_T, \sigma^2_T)$ *be a partially reduced grid-polyhedron domain where, for all* $\mathcal{H} \in \mathbb{GP}_n$*, there exists* $\mathcal{L}' \in \mathbb{G}_n$ *and* $\mathcal{P}' \in \mathbb{CP}_n$ *such that*

$$\mathcal{L}' := \sigma^1_T(\mathcal{L}, \mathcal{P}) = \sigma^1_=(\mathcal{L}, \mathcal{P}),$$
$$\mathcal{P}' := \sigma^2_T(\mathcal{L}, \mathcal{P}) = \sigma^2_=(\mathcal{L}, \mathcal{P})$$



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P}')$.

Figure 6.4: Two Grid-Polyhedra.

and for some constraint system $\mathcal{C}_{\mathcal{P}}{}'$ for $\mathcal{P}'$, for all $\big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d\big) \in \mathcal{C}_{\mathcal{P}}{}'$, there exists a point $\mathbf{w} \in \mathcal{H}$ such that $\langle \mathbf{v}, \mathbf{w} \rangle = d$. Then $(\mathbb{GP}_n, \sigma_T^1, \sigma_T^2)$ is a tight *product domain and we say that $\mathcal{C}_{\mathcal{P}}{}'$ is a tight polyhedron constraint system for $\mathcal{H}$.*

If $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ and $\sigma_T^1(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma_T^2(\mathcal{L}, \mathcal{P}) = \mathcal{P}$, then we say that the pair $(\mathcal{L}, \mathcal{P})$ is a tight product.

Observe that in Definitions 6.15 and 6.16 we do not require the constraint system $\mathcal{C}_{\mathcal{P}}{}'$ to be in minimal form. This is due to the fact that in Chapter 7 when we consider the combination of a grid with a bounded difference shape or octagon we will want to look at all possible constraints in a closed constraint system including those which may be redundant.

Example 6.17 *Consider the grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$ where $\mathcal{C}_{\mathcal{L}} := \{x \equiv_3 2,\ y \equiv_2 0\}$ and the polyhedron $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_{\mathcal{P}} := \{x \leq 10,\ 3 \leq y \leq 9,\ 6 \leq x + y,\ -6 \leq x - y \leq 4\}.$$

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, which is shown in Figure 6.4(a). Then it can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight polyhedron constraint system for $\mathcal{H}$ and also note that $\mathcal{C}_{\mathcal{P}}$ is not a weakly tight constraint system for $\mathcal{H}$ as some of the constraints are not saturated by any point of $\mathcal{L}$, for example $y \leq 9$. Now consider the grid $\mathcal{L}$ together with the polyhedron $\mathcal{P}' = \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$ in $\mathbb{CP}_2$ where

$$\mathcal{C}_{\mathcal{P}}{}' := \{2 \leq x \leq 8,\ 4 \leq y \leq 8\}.$$

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}')$, which is shown in Figure 6.4(b). Then not only can it be seen that $\mathcal{C}_{\mathcal{P}}{}'$ is a tight polyhedron constraint system for $\mathcal{H}$ as every constraint is saturated by at least one grid-polyhedra point, but also $(\mathcal{L}, \mathcal{P}')$ is a reduced product.

We will show that there are two main ways to improve the polyhedron constraint system so that the grid-polyhedron is a tight or weakly tight pair. One way is to move in the bounding hyperplanes of the polyhedron component so that they are closer to the grid-polyhedron points, this is discussed in Section 6.4.1. Another way is to add new constraints to the polyhedron representation, this will be discussed in Section 6.6.

6.4.1 Weakly Tight Operations

An operation to produce a weakly tight grid-polyhedron will allow us to move in the bounding hyperplanes of the polyhedron component so that they contain at least one point in the grid component. Therefore we could possibly improve the polyhedron representation without adding extra constraints. Example 6.4 on Page 88 illustrated why an operation that can shrink a polyhedron with respect to a grid is an important operation as it can lead to a canonical form.

To move in the existing polyhedron constraints we must try to find congruences for the grid which will be parallel to the constraint in the polyhedron representation. The aim is that if we have a constraint $\langle \mathbf{v}, \mathbf{x} \rangle \leq d$ we can take the directional vector $\mathbf{v}$ and produce a congruence equation that will have solutions parallel to the constraint, thus we will have a measure of how much we can move the constraint bound.

Definition 6.18 (Directed Non-Redundant Congruence.) *Let $\mathcal{L} \in \mathbb{G}_n$, $\mathbf{v} \in \mathbb{Q}^n$, $f \in \mathbb{Q}_+$ and $d \in \mathbb{Q}$. Then we say that $c = \big(\langle \mathbf{v}, \mathbf{x} \rangle \equiv_f d\big)$ is a* directed non-redundant congruence (dnc) *for $\mathcal{L}$ and $\mathbf{v}$ if $\mathcal{L} \subseteq \mathrm{gcon}\big(\{c\}\big)$ and, for all $s \in \mathbb{Z}$, if $c_s = \big(\langle \mathbf{v}, \mathbf{x} \rangle = d + s \cdot f\big)$, $\mathcal{L} \cap \mathrm{gcon}\big(\{c_s\}\big) \neq \varnothing$.*

Lemma 6.19 *If $\mathcal{C}_\mathcal{L}$ is a congruence system in minimal form and $c = \big(\langle \mathbf{v}, \mathbf{x} \rangle \equiv_f d\big) \in \mathcal{C}_\mathcal{L}$, then c is a dnc for the grid $\mathrm{gcon}(\mathcal{C}_\mathcal{L})$ and $\mathbf{v}$.*

Proof. Since $c \in \mathcal{C}_\mathcal{L}$, we have $\mathcal{L} \subseteq \mathrm{gcon}\big(\{c\}\big)$. Let $s \in \mathbb{Z}$ and $c_s = \big(\langle \mathbf{v}, \mathbf{x} \rangle = d + s \cdot f\big)$. Let ${\mathcal{C}_\mathcal{L}}' = \big(\mathcal{C}_\mathcal{L} \setminus \{c\}\big) \cup \{c_s\}$; then as $\mathcal{C}_\mathcal{L}$ is in minimal form, ${\mathcal{C}_\mathcal{L}}'$ is also in minimal form. By Lemma 3.16, ${\mathcal{C}_\mathcal{L}}'$ is consistent so that $\mathrm{gcon}({\mathcal{C}_\mathcal{L}}') = \mathcal{L} \cap \mathrm{gcon}\big(\{c_s\}\big) \neq \varnothing$. □

Example 6.20 *Consider the grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L}) \in \mathbb{G}_n$, where*

$$\mathcal{C}_\mathcal{L} = \{x \equiv_2 0,\ y \equiv_3 0\}.$$

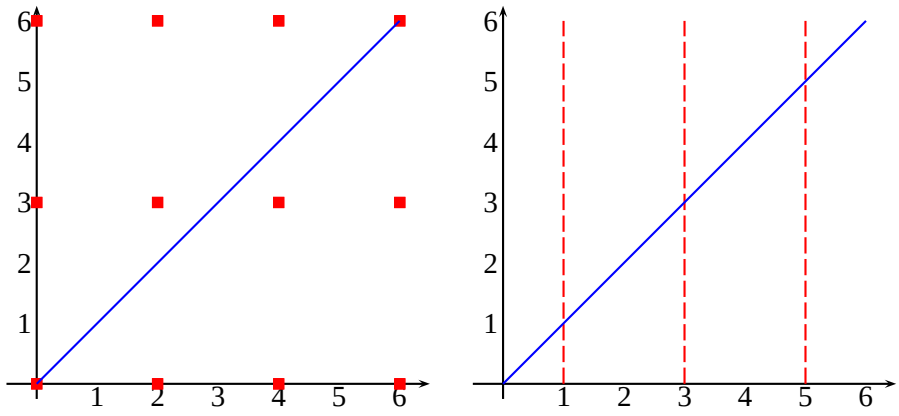
$\mathcal{L}$ is illustrated by the points in Figure 6.5(a), and the constraint $x - y = 0$, is illustrated by the diagonal line in Figure 6.5(a). Then taking $\mathbf{v} = (1, -1)$ we can see that the congruence

$$x - y \equiv_1 0$$

is a dnc for $\mathcal{L}$ and $\mathbf{v}$.

Now consider the grid $\mathcal{L}' = \mathrm{gcon}({\mathcal{C}_\mathcal{L}}') \in \mathbb{G}_n$, where

$${\mathcal{C}_\mathcal{L}}' = \{x \equiv_2 1\}.$$



(a) Grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ and vector $\mathbf{v}$. (b) Grid $\mathcal{L}' = \mathrm{gcon}(\mathcal{C}_\mathcal{L}')$ and vector $\mathbf{v}$.

Figure 6.5: We can create a dnc for some $\mathcal{L}$ and $\mathbf{v}$, but not all.

$\mathcal{L}'$ is illustrated by the dashed lines in Figure 6.5(b), and the constraint $x - y = 0$, is illustrated by the diagonal line in Figure 6.5(b). Then taking $\mathbf{v} = (1, -1)$ we can see that there is no dnc for $\mathcal{L}'$ and $\mathbf{v}$, since in $\mathcal{L}'$ the variable y can take any value.

Algorithm 2: The directed non-redundant congruence algorithm.
Input: A congruence system $\mathcal{C}_\mathcal{L}$ in $\mathbb{Q}^n$ in minimal form and a vector $\mathbf{v} \in \mathbb{Q}^n$.
Output: A triple (bool, m, t) where $\mathrm{bool} \in \{\mathrm{true}, \mathrm{false}\}$, $m \in \mathbb{Q}_+$ and $t \in \mathbb{Q}$.

(1) $t := 0, m := 0$, $\mathbf{w} = \mathbf{0}$
(2) **for** $i = n$ **to** 1
(3) **if** $v_i \neq w_i$
(4) **if** $\beta = \big(\langle \mathbf{a}, \mathbf{x}\rangle \equiv_f b\big) \in \mathcal{C}_\mathcal{L}$. $\mathrm{piv}_<(\beta) = i$
(5) $u := \frac{v_i - w_i}{a_i}$
(6) $m := \gcd(m, u \cdot f)$
(7) $t := t + u \cdot b$
(8) $\mathbf{w} := \mathbf{w} + u \cdot \mathbf{a}$
(9) **else**
(10) **return** $(\mathrm{false}, 0, 0)$
(11) **return** (true, m, t)

Algorithm 2 provides a method for computing a dnc for a given grid and vector.

Proposition 6.21 *Given $\mathbf{v} \in \mathbb{Q}^n$ and a non-empty grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$, where $\mathcal{C}_\mathcal{L}$ is in minimal form, suppose* Algorithm 2 *returns the triple (bool, m, t). If $\mathit{bool} = \mathit{true}$, then $c = \big(\langle \mathbf{v}, \mathbf{x}\rangle \equiv_m t\big)$ is an dnc for $\mathcal{L}$ and $\mathbf{v}$ and if $\mathit{bool} = \mathit{false}$, then for all $s \in \mathbb{Q}$, $\mathcal{L} \cap \mathrm{con}\Big(\big\{\langle \mathbf{v}, \mathbf{x}\rangle = s\big\}\Big) \neq \varnothing$.*

To prove this proposition, we first prove an invariant property of the loop on line (2) in Algorithm 2.

Lemma 6.22 *Suppose that, given a non-empty grid $\mathcal{L} \in \mathbb{G}_n$ and a vector $\mathbf{v}$, Algorithm 2 executes line* (2) *j times. Let m_j be the value of m, t_j the value of t and $\mathbf{w}_j$ the value of $\mathbf{w}$ at the end of the j-th execution of lines* (3) *to* (10)*. Let also $c_j = \big(\langle \mathbf{w}_j, \mathbf{x}\rangle \equiv_{m_j} t_j\big)$. Then, if line (10) is not executed, c_j is an dnc for $\mathcal{L}$ and $\mathbf{w}_j$ and $\mathbf{w}_j = (w_1, \ldots, w_{n-j-1}, v_{n-j}, \ldots, v_n)$, for some $w_1, \ldots, w_{n-j-1} \in \mathbb{Q}$.*

Proof. Suppose $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$, where $\mathcal{C}_\mathcal{L}$ is in minimal form. We prove the result holds by induction on j. The base case is trivial since in this case $j = 0$ and $\mathbf{w}_0 = \mathbf{0}$, $t_0 = 0$ and $m_0 = 0$.

Suppose now that $j > 0$ and let $i = n - j$. By the induction hypothesis, given any non-empty grid $\mathcal{L}$, then, after the $(j-1)$-th execution of lines (3) to (10), c_{j-1} is an dnc for $\mathcal{L}$ and $\mathbf{w}_{j-1} = (w_1, \ldots, w_i, v_{i+1}, \ldots, v_n)$, for some $w_1, \ldots, w_i \in \mathbb{Q}$. If the tests on line (3) fails, then $c_j = c_{j-1}$ and $\mathbf{w}_j = \mathbf{w}_{j-1}$. Suppose now that the tests on line (3) succeeds. By line (8), as $\mathrm{piv}_<(\beta) = i$, we also have $\mathbf{w}_j = (w_1, \ldots, w_{i-1}, v_i, \ldots, v_n)$, for some $w_1, \ldots, w_{i-1} \in \mathbb{Q}$.

Suppose $s \in \mathbb{Z}$ and $\gamma_j := \big(\langle \mathbf{w}_j, \mathbf{x}\rangle = t_j + s \cdot m_j\big)$, then, by Definition 6.18, we need to prove:

$$\mathcal{L} \subseteq \mathrm{gcon}\big(\{c_j\}\big); \tag{6.1}$$

$$\mathcal{L} \cap \mathrm{gcon}\big(\{\gamma_j\}\big) \neq \varnothing. \tag{6.2}$$

Note that, by lines (6), (7) and (8),

$$c_j = \big(\langle \mathbf{w}_{j-1} + u \cdot \mathbf{a}, \mathbf{x}\rangle \equiv_{m_j} t_{j-1} + u \cdot b\big)$$

and $m_j = \gcd(m_{j-1}, u \cdot f)$.

By Definition 6.18 and since c_{j-1} is a dnc for $\mathcal{L}$ and $\mathbf{w}_{j-1}$, $\mathcal{L} \subseteq \mathrm{gcon}\big(\{c_{j-1}\}\big)$; also, by line (4), $\beta \in \mathcal{C}_\mathcal{L}$; thus property (6.1) holds. We now prove that property (6.2) holds. By line (6), there exist $p, q \in \mathbb{Z}$ such that $p \cdot m_{j-1} + q \cdot (u \cdot f) = m_j$. Let

$$\beta_e = \big(\langle \mathbf{a}, \mathbf{x}\rangle = b + s \cdot q \cdot f\big),$$
$$\gamma_e = \big(\langle \mathbf{w}_{j-1}, \mathbf{x}\rangle = t_{j-1} + s \cdot p \cdot m_{j-1}\big).$$

By Lemma 6.19, β is an dnc for $\mathcal{L}$ and $\mathbf{a}$ so that, by Definition 6.18, $\mathcal{L}' := \mathcal{L} \cap \mathrm{gcon}\big(\{\beta_e\}\big) \neq \varnothing$. Consider the congruence system $\mathcal{C}_\mathcal{L}{}' = \big(\mathcal{C}_\mathcal{L} \setminus \{\beta\}\big) \cup \{\beta_e\}$; then, as $\mathrm{piv}_<(\beta_e) = \mathrm{piv}_<(\beta) = i$ and $\mathcal{C}_\mathcal{L}$ is in minimal form, $\mathcal{C}_\mathcal{L}{}'$ is in minimal form. Moreover, since $\mathrm{gcon}\big(\{\beta_e\}\big) \subseteq \mathrm{gcon}\big(\{\beta\}\big)$, $\mathcal{L}' = \mathrm{gcon}(\mathcal{C}_\mathcal{L}{}')$. Also, for both grids $\mathcal{L}$ and $\mathcal{L}'$, for the first $j-1$ iterations of the loop, line (4) will select exactly the same congruences; and hence, at the end of the $(j-1)$-th iteration, m_{j-1}, t_{j-1} and $\mathbf{w}_{j-1}$ will be the values of m, t and $\mathbf{w}$, respectively, when using $\mathcal{C}_\mathcal{L}{}'$ instead of $\mathcal{C}_\mathcal{L}$. Thus, by the inductive hypothesis, c_{j-1} is also an dnc for $\mathcal{L}'$ and $\mathbf{w}_{j-1}$. By Definition 6.18, $\mathcal{L}' \cap \mathrm{gcon}\big(\{\gamma_e\}\big) \neq \varnothing$, so that, since $\mathcal{L}' = \mathcal{L} \cap \mathrm{gcon}\big(\{\beta_e\}\big) \subseteq \mathcal{L}$, we have $\mathcal{L} \cap \mathrm{gcon}\big(\{\beta_e, \gamma_e\}\big) \neq \varnothing$. As γ_j is the sum of equalities $u \cdot \beta_e$ and γ_e, we must also have $\mathcal{L} \cap \mathrm{gcon}\big(\{\gamma_j\}\big) \neq \varnothing$ and property (6.2) holds. □

Proof [of Proposition 6.21.] Suppose that the algorithm executes lines (6) to (8) k times so that $0 \leq k \leq n$. For $0 \leq j \leq k$, let m_j be the value of m, t_j the value of t and $\mathbf{w}_j$ the value of $\mathbf{w}$ at the end of j-th iteration.

Suppose first that bool $=$ true is returned by the algorithm. Then line (10) is not executed, so that, by Lemma 6.22, letting $j = n$, $\mathbf{w}_n = \mathbf{v}$ and the result follows.

Suppose now that bool $=$ false is returned. This means that $k \geq 1$ and, in the k-th iteration, the test on line (3) succeeded so that $w_{n-k} \neq v_{n-k}$; and the test on line (4) failed so that there is no congruence $\beta \in \mathcal{C}_{\mathcal{L}}$ such that $\mathrm{piv}_{<}(\beta) = n - k$. By Lemma 6.22, $c_{k-1} := \big(\langle \mathbf{w}_{k-1}, \mathbf{x}\rangle \equiv_{m_{k-1}} t_{k-1}\big)$ is a dnc for $\mathcal{L}$ and $\mathbf{w}_{k-1} = (w_1, \ldots, w_{n-k}, v_{n-k+1}, \ldots, v_n)$, for some $w_1, \ldots, w_{n-k} \in \mathbb{Q}$. Since we have $w_{n-k} \neq v_{n-k}$, $\mathrm{piv}_{<}(\mathbf{w}_{k-1} - \mathbf{v}) = n - k$. Suppose, by contraposition, that there exists $s \in \mathbb{Q}$ such that $\mathcal{L} \cap \mathrm{con}\Big(\big\{\langle \mathbf{v}, \mathbf{x}\rangle = s\big\}\Big) = \varnothing$; let $\mathcal{C}_{\mathcal{L}}{}' := \mathcal{C}_{\mathcal{L}} \cup \big\{\langle \mathbf{w}_{k-1} - \mathbf{v}, \mathbf{x}\rangle = t_{k-1} - s\big\}$ and $\mathcal{L}' := \mathrm{gcon}(\mathcal{C}_{\mathcal{L}}{}')$. Then, as there is no congruence $\beta \in \mathcal{C}_{\mathcal{L}}$ such that $\mathrm{piv}_{<}(\beta) = n - k$, $\mathcal{C}_{\mathcal{L}}{}'$ is in minimal form and, by Lemma 3.16, the grid $\mathcal{L}'$ is non-empty so that Lemma 6.22 can be applied to $\mathcal{L}'$. Moreover, starting with $\mathcal{C}_{\mathcal{L}}{}'$, for the first $k - 1$ iterations of the loop, line (4) will select exactly the same congruences as those selected when starting with $\mathcal{L}$; and hence, m_{k-1}, t_{k-1} and $\mathbf{w}_{k-1}$ will also be the values of m, t and $\mathbf{w}$, respectively, at the end of the $(k-1)$-th iteration when using $\mathcal{C}_{\mathcal{L}}{}'$ instead of $\mathcal{C}_{\mathcal{L}}$. Thus, by Lemma 6.22, letting $j = k-1$, $\big(\langle \mathbf{w}_{k-1}, \mathbf{x}\rangle \equiv_{m_{k-1}} t_{k-1}\big)$ is also a dnc for $\mathcal{L}'$ and $\mathbf{w}_{k-1}$ so that, by Definition 6.18, we obtain

$$
\begin{aligned}
\varnothing \neq \mathcal{L}' \cap \mathrm{gcon}\Big(\big\{\langle \mathbf{w}_{k-1}, \mathbf{x}\rangle = t_{k-1}\big\}\Big) \\
&= \mathcal{L} \cap \mathrm{gcon}\Big(\big\{\langle \mathbf{w}_{k-1}, \mathbf{x}\rangle = t_{k-1}, \langle \mathbf{w}_{k-1} - \mathbf{v}, \mathbf{x}\rangle = t_{k-1} - s\big\}\Big) \\
&= \mathcal{L} \cap \mathrm{gcon}\Big(\big\{\langle \mathbf{w}_{k-1}, \mathbf{x}\rangle = t_{k-1}, \langle \mathbf{v}, \mathbf{x}\rangle = s\big\}\Big) \\
&\subseteq \mathcal{L} \cap \mathrm{gcon}\Big(\big\{\langle \mathbf{v}, \mathbf{x}\rangle = s\big\}\Big).
\end{aligned}
$$

Hence $\mathcal{L} \cap \mathrm{gcon}\Big(\big\{\langle \mathbf{v}, \mathbf{x}\rangle = s\big\}\Big) \neq \varnothing$ which is a contradiction. □

As this algorithm assumes the congruence system for the grid $\mathcal{L}$ is in minimal form the complexity of Algorithm 2 is $\mathrm{O}\big(n^2\big)$ if $\mathcal{L}$ is not rectilinear. If $\mathcal{L}$ is rectilinear then the complexity of Algorithm 2 is linear in the number of non-zero coefficients in $\mathbf{v}$. Let $\mathrm{DNC} : \mathbb{G}_n \times \mathbb{R}^n \to \mathbb{R} \times \mathbb{R}$ be the partial function such that $\mathrm{DNC}(\mathcal{L}, \mathbf{v})$ is the output of Algorithm 2 if bool $=$ true. Given an algorithm for generating directed non-redundant congruences, Algorithm 3 shows how to move each of the constraint bounds so that the constraint system for $\mathcal{P}$ is a weakly tight for $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. As any equality $\big(\langle \mathbf{v}, \mathbf{x}\rangle = d\big)$ can be represented by the two inequalities $\big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big)$ and $\big(\langle \mathbf{v}, \mathbf{x}\rangle \geq d\big)$, Algorithm 3 and Proposition 6.23 will assume that $\mathcal{C}_{\mathcal{P}}$ is a set of inequalities.

Algorithm 3: The weakly tight constraint system algorithm.
Input: $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ and $\mathcal{L}$.
Output: The constraint system $\mathcal{C}_\mathcal{P}{}'$.

(1) $\mathcal{C}_\mathcal{P}{}' := \varnothing$
(2) **while** $\nu = \big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big) \in \mathcal{C}_\mathcal{P}$
(3) **if** $\mathrm{DNC}(\mathcal{L}, \mathbf{v}) = (m, t)$
(4) $d' := d - \big((d - t) \bmod m\big)$
(5) $\mathcal{C}_\mathcal{P}{}' := \mathcal{C}_\mathcal{P}{}' \cup \Big\{\big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d'\big)\Big\}$
(6) **else**
(7) $\mathcal{C}_\mathcal{P}{}' := \mathcal{C}_\mathcal{P}{}' \cup \{\nu\}$
(8) $\mathcal{C}_\mathcal{P} := \mathcal{C}_\mathcal{P} \setminus \{\nu\}$
(9) **return** $\mathcal{C}_\mathcal{P}{}'$

As Algorithm 2 has complexity $\mathrm{O}\big(n^2\big)$ and the DNC function, which uses Algorithm 2, is performed μ times, where $\#\,\mathcal{C}_\mathcal{P} = \mu$, Algorithm 3 has complexity $\mathrm{O}\big(n^2\mu\big)$.

Proposition 6.23 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a constraint product and let $\mathcal{C}_\mathcal{P}{}'$ be the constraint system returned by* Algorithm 3. *Then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}')\big)$ is a weakly tight product.*

Proof. Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$, $\mathcal{C}_\mathcal{P}$ be a constraint system with no equalities such that $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ and $\mathcal{C}_\mathcal{L}$ be a congruence system in minimal form such that $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$. Since at the end of each iteration of the while loop $\#(\mathcal{C}_\mathcal{P})$ is reduced by one, Algorithm 3 will terminate and hence is an algorithm. Let $\mathcal{C}_i, \mathcal{C}_i'$ and d_i' denote the values computed for $\mathcal{C}_\mathcal{P}, \mathcal{C}_\mathcal{P}{}'$ and d', respectively, at the end of the i-th iteration of the while loop. Then we will show that

1. $\mathcal{H} := \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_i \cup \mathcal{C}_i')\big)$;
2. $\mathcal{H}_i' := \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_i')\big)$ is a weakly tight product.

Initially (1) and (2) hold since $\mathcal{C}_\mathcal{P} = \mathcal{C}_0$ and $\mathcal{C}_\mathcal{P}{}' = \mathcal{C}_0' = \varnothing$. We now assume that (1) and (2) hold for $i - 1$ iterations of the while loop where $i \geq 1$. On line (2) of Algorithm 3, the constraint $\nu = \big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big) \in \mathcal{C}_\mathcal{P}$ is selected. There are 3 cases to consider for ν, $\mathrm{DNC}(\mathcal{L}, \mathbf{v}) = (m, t)$ where $m \neq 0$, $\mathrm{DNC}(\mathcal{L}, \mathbf{v}) = (0, t)$ and $\mathrm{DNC}(\mathcal{L}, \mathbf{v})$ is undefined.

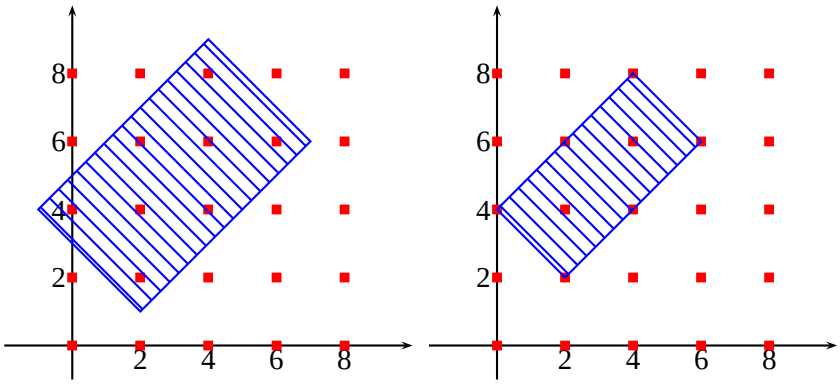
First let us suppose that $\mathrm{DNC}(\mathcal{L}, \mathbf{v}) = (m, t)$ where $m \neq 0$. On line (4),

$$d' = d - \big((d - t) \bmod m\big),$$

so by the definition of $\bmod$ in Section 2.1,

$$d - m < d' \leq d \text{ and } 0 \leq (d - t) \bmod m < m.$$

Hence, for some $s \in \mathbb{Z}$, $d' = t + s \cdot m$. Let $\nu' = \big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d'\big)$ and $\nu_e' = \big(\langle \mathbf{v}, \mathbf{x}\rangle = d'\big)$. By Proposition 6.21, $\beta = \big(\langle \mathbf{v}, \mathbf{x}\rangle \equiv_m t\big)$ is a dnc for $\mathcal{L}$ and $\mathbf{v}$. By Definition 6.18, $\mathcal{L} \subseteq \mathrm{gcon}\big(\{\beta\}\big)$



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$.

Figure 6.6: Moving constraints for a grid-polyhedron.

and $\mathcal{L} \cap \mathrm{gcon}\big(\{\nu_e'\}\big) \neq \varnothing$. Therefore, since $\mathcal{H}'_{i-1}$ is weakly tight, we have that $\mathcal{H}'_i$ is weakly tight. Also since $\mathcal{H} \subseteq \mathrm{gcon}\big(\{\beta\}\big)$, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_i \cup \mathcal{C}'_i)\big)$.

Now suppose $\mathrm{DNC}(\mathcal{L}, \mathbf{v}) = (0, t)$. Then by Proposition 6.21, $\beta = \big(\langle \mathbf{v}, \mathbf{x} \rangle = t\big)$ is a dnc for $\mathcal{L}$ and $\mathbf{v}$. As $\mathcal{H}$ is a constraint product $d = t$ and on line (4),

$$d' = d - \big((d - t) \bmod m\big) = t.$$

Therefore, since $\mathcal{H}'_{i-1}$ is weakly tight, we have that $\mathcal{H}'_i$ is weakly tight. Also since $\mathcal{H} \subseteq \mathrm{gcon}\big(\{\beta\}\big)$, $\nu = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d'\big)$ and $\nu \in \mathcal{C}_{i-1}$, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_i \cup \mathcal{C}'_i)\big)$.

Finally suppose that $\mathrm{DNC}(\mathcal{L}, \mathbf{v})$ is undefined. Then, by Proposition 6.21, we have that $\mathcal{L} \cap \mathrm{con}\Big(\big\{\langle \mathbf{v}, \mathbf{x} \rangle = s\big\}\Big) \neq \varnothing$ for all $s \in \mathbb{Q}$, so $\mathcal{L} \cap \mathrm{con}\big(\{\nu\}\big) \neq \varnothing$. Therefore, since $\mathcal{H}'_{i-1}$ is weakly tight, we have that $\mathcal{H}'_i$ is weakly tight. Also since $\nu \in \mathcal{C}_{i-1}$, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_i \cup \mathcal{C}'_i)\big)$.

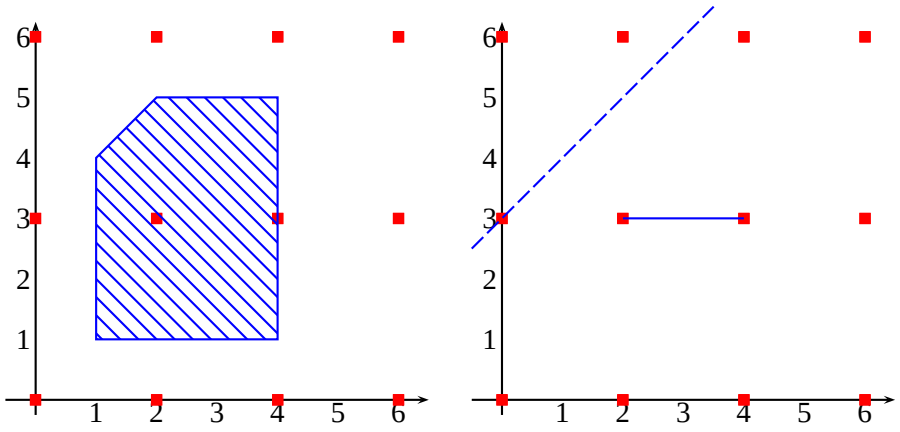
Therefore if $\mathcal{C}_{\mathcal{P}}{}'$ is the constraint system returned by Algorithm 3, then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is a weakly tight product. □

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a constraint product. Then if $\mathcal{C}_{\mathcal{P}}{}'$ is the constraint system returned by Algorithm 3, $\sigma^1_W(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma^2_W(\mathcal{L}, \mathcal{P}) = \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$.

Example 6.24 *Consider the grid-polyhedron $\mathcal{H} = \mathcal{L} \cap \mathcal{P}$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ and $\mathcal{C}_{\mathcal{L}} := \{x \equiv_2 0,\ y \equiv_2 0\}$ and $\mathcal{P}$ is given by the constraint system*

$$\mathcal{C}_{\mathcal{P}} := \{3 \leq x + y \leq 13,\ -5 \leq x - y \leq 1\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 6.6(a). From Algorithm 2 *and* Algorithm 3 *described above we can calculate the modulus and inhomogeneous terms to produce the directed non-redundant congruences $x + y \equiv_2 0$ and $x - y \equiv_2 0$ and we can compute the new constraint bounds.*



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$.

Figure 6.7: Algorithm 3 does not improve redundant constraints.

Therefore we get the new constraint system $\mathcal{C}_{\mathcal{P}}{}'$ where

$$\mathcal{C}_{\mathcal{P}}{}' := \{4 \leq x + y \leq 12,\ -4 \leq x - y \leq 0\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ can be seen in Figure 6.6(b). It shows that not only is the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ a tight product, but also $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is a reduced product.

Example 6.25 shows that Algorithm 3 can move in the constraint bounds but it does not improve the constraint bounds so that they are moved in with respect to the other constraint bounds. That is, Algorithm 3 will not remove or improve redundant constraints.

Example 6.25 *Consider the grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ and $\mathcal{C}_{\mathcal{L}} := \{x \equiv_2 0,\ y \equiv_3 0\}$ and $\mathcal{P}$ is given by the constraint system*

$$\mathcal{C}_{\mathcal{P}} := \{1 \leq x \leq 4,\ 1 \leq y \leq 5,\ -3 \leq x - y\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 6.7(a). From Algorithm 2 *and* Algorithm 3 *described above we can calculate the modulus and inhomogeneous terms to produce the directed non-redundant congruences $x \equiv_2 0, y \equiv_3 0$ and $x - y \equiv_1 0$, and we can compute the new constraint bounds. Therefore we get the new constraint system $\mathcal{C}_{\mathcal{P}}{}'$ where*

$$\mathcal{C}_{\mathcal{P}}{}' := \{2 \leq x \leq 4,\ y = 3,\ -3 \leq x - y\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ can be seen in Figure 6.7(b). Now the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is reduced product, but it can be seen in Figure 6.7(b) that the constraint $-3 \leq x - y$, illustrated by the dashed line, is now redundant.

6.4.2 Emptiness

To test if a grid-polyhedron is empty we need to test if the polyhedron contains any grid points. To do this we first check that each component of the grid-polyhedron is non-empty. Before we introduce the test for emptiness we first give the definition of a paired constraint system. As any equality $\big(\langle \mathbf{v}, \mathbf{x}\rangle = d\big)$ can be represented by the two inequalities $\big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big)$ and $\big(\langle \mathbf{v}, \mathbf{x}\rangle \geq d\big)$, for Section 6.4.2 we will assume that $\mathcal{C}_\mathcal{P}$ is a set of inequalities.

Definition 6.26 (Paired Constraint System.) *Let $\mathcal{C}_\mathcal{P}$ be a consistent constraint system in $\mathbb{R}^n$ and let $\mathcal{P} = \operatorname{con}(\mathcal{C}_\mathcal{P}) \in \mathbb{P}_n$. Then $\mathcal{C}_\mathcal{P}$ is a* paired constraint system *if, for each constraint $\nu = \big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big) \in \mathcal{C}_\mathcal{P}$:*

1. *there exists a point $\mathbf{p} \in \mathcal{P}$ such that $\langle \mathbf{v}, \mathbf{p}\rangle = d$;*
2. *if $\mathcal{P}$ is bounded in the direction $-\mathbf{v}$, then there exists $\nu' = \big(\langle -\mathbf{v}, \mathbf{x}\rangle \leq d'\big) \in \mathcal{C}_\mathcal{P}$.*

Given a constraint system $\mathcal{C}_\mathcal{P}$, we can create a paired constraint system $\mathcal{C}_{\leq\leq}$ such that $\operatorname{con}(\mathcal{C}_\mathcal{P}) = \operatorname{con}(\mathcal{C}_{\leq\leq})$ by applying the simplex algorithm [67, 76]. We will use the paired constraint system in the test for emptiness. Given a grid $\mathcal{L}$ and $\mathcal{C}_{\leq\leq}$ we can apply Algorithm 3 to $\mathcal{H} = \big(\mathcal{L}, \operatorname{con}(\mathcal{P}_{\leq\leq})\big)$ and get the constraint system $\mathcal{C}_{\leq\leq}{}'$, which is weakly tight for $\mathcal{H}$. Then if $\mathcal{C}_{\leq\leq}{}'$ is inconsistent then we know $\mathcal{H} = \big(\mathcal{L}, \operatorname{con}(\mathcal{C}_\mathcal{P})\big)$ is empty since $\operatorname{con}(\mathcal{C}_\mathcal{P}) = \operatorname{con}(\mathcal{C}_{\leq\leq}) = \operatorname{con}(\mathcal{C}_{\leq\leq}{}')$. If $\mathcal{C}_{\leq\leq}{}'$ is consistent then we don't know if $\mathcal{H} = \big(\mathcal{L}, \operatorname{con}(\mathcal{C}_\mathcal{P})\big)$ is empty.

Proposition 6.27 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ where $\mathcal{P} = \operatorname{con}(\mathcal{C}_\mathcal{P})$ is a paired constraint system and $\mathcal{C}_\mathcal{P}{}'$ is the constraint system returned after applying* Algorithm 3 *to $\mathcal{H} = \big(\mathcal{L}, \operatorname{con}(\mathcal{C}_\mathcal{P})\big)$. Suppose $\mathcal{C}_\mathcal{P}{}'$ is inconsistent. Then $\mathcal{H} = \varnothing$.*

Proof. The result follows from Proposition 6.23. □

As Algorithm 3 has complexity $\mathrm{O}\big(n^2\mu\big)$, where $\#\,\mathcal{C}_\mathcal{P} = \mu$, if $\mathcal{C}_\mathcal{P}$ is a paired constraint system then the test for emptiness has complexity $\mathrm{O}\big(n^2\mu\big)$. Otherwise the complexity is that of computing the paired constraint system.

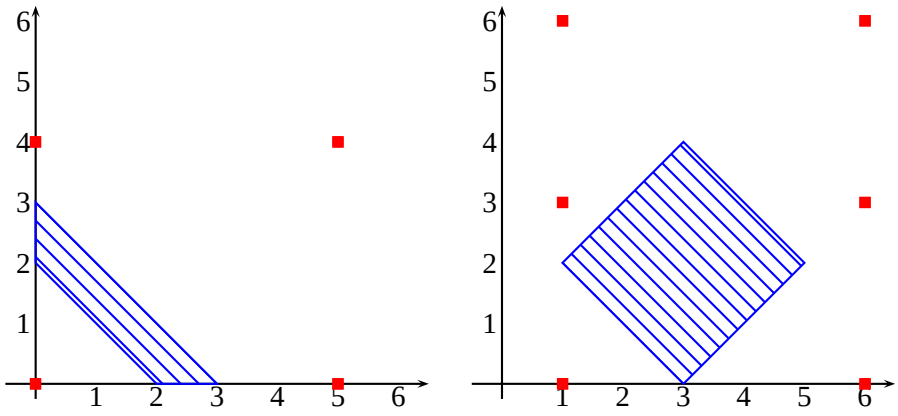
Example 6.28 shows this method succeeding. Unfortunately, if the grid-polyhedron $\mathcal{H}$ is not a reduced product it is possible that $\mathcal{H}$ is empty but this method does not detect this. Example 6.28 also illustrates this.

Example 6.28 *Consider the grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} = \operatorname{gcon}(\mathcal{C}_\mathcal{L})$ and $\mathcal{C}_\mathcal{L} := \{x \equiv_5 0,\ y \equiv_4 0\}$ and $\mathcal{P}$ is given by the constraint system*

$$\mathcal{C}_\mathcal{P} := \{0 \leq x,\ 0 \leq y,\ 2 \leq x + y \leq 3\}.$$

It can be seen in Figure 6.8(a) that $(\mathcal{L}, \mathcal{P})$ is a weakly tight product. In this case the paired representation, $\mathcal{C}_{\leq\leq}$, of $\mathcal{C}_\mathcal{P}$ is given by

$$\mathcal{C}_{\leq\leq} := \{0 \leq x \leq 3,\ 0 \leq y \leq 3,\ 2 \leq x + y \leq 3\}.$$



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H}' = (\mathcal{L}', \mathcal{P}')$.

Figure 6.8: The emptiness test succeeds and fails.

From Algorithm 2 *described above we can calculate the modulus and inhomogeneous terms to produce the directed non-redundant congruence* $x + y \equiv_1 0$. *Applying* Algorithm 3 *to* $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\leq\leq})\big)$ *we get the constraint system*

$$\mathcal{C}_{\leq\leq}{}' := \{0 \leq x \leq 0,\ 0 \leq y \leq 0,\ 2 \leq x + y \leq 3\}.$$

As $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\leq\leq}{}')$ *and* $\mathcal{C}_{\leq\leq}{}'$ *is inconsistent, we know that* $\mathcal{H} = \varnothing$.

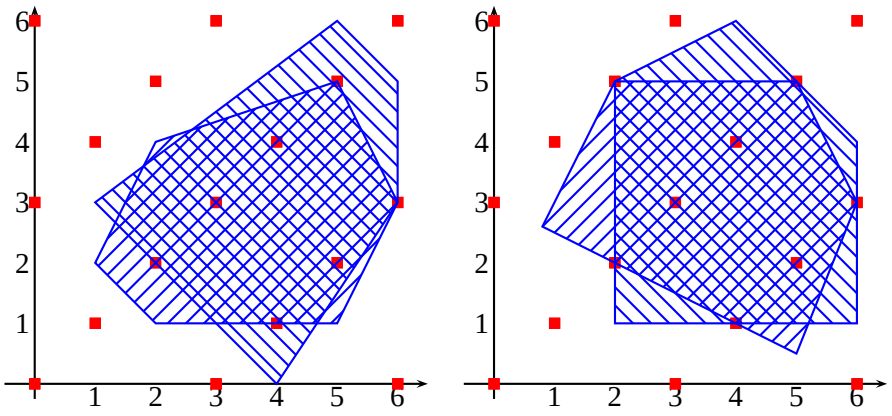
Now consider the grid-polyhedron $\mathcal{H}' = (\mathcal{L}', \mathcal{P}')$, *where* $\mathcal{L}' = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ *and* $\mathcal{C}_{\mathcal{L}} := \{x \equiv_5 1,\ y \equiv_3 0\}$ *and* $\mathcal{P}'$ *is given by the constraint system*

$$\mathcal{C}_{\mathcal{P}}{}' := \{3 \leq x + y \leq 7,\ -1 \leq x - y \leq 3\}.$$

$\mathcal{H}'$ *can be seen in Figure 6.8(b). In this case* $\mathcal{C}_{\mathcal{P}}{}' = \mathcal{C}_{\leq\leq}{}'$. *From* Algorithm 2 *described above we can calculate the modulus and inhomogeneous terms to produce the directed non-redundant congruences* $x + y \equiv_1 0, x - y \equiv_1 0$. *Therefore* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\leq\leq})\big)$ *is a weakly tight product,* $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\leq\leq})\big)$ *and as* $\mathcal{C}_{\leq\leq}$ *is consistent we do not know if the grid-polyhedron is empty.*

6.5 The Grid-Polyhedron Domain Operations

We will now consider each of the abstract operations, such as those based on the set-theoretic operations and also affine image, affine pre-image and widening. We will consider each operation for a partially reduced product and show if each operation will preserve the given reduction. For example, if we have grid-polyhedra which are reduced products we will show if after the operation is applied whether or not the resulting grid-polyhedron is still a reduced product.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedra $\mathcal{H}'_1$ and $\mathcal{H}'_2$.

Figure 6.9: The comparison and equality test returning the result "don't know" for relational grid-polyhedra.

6.5.1 Comparison

For any pair of grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$, $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_n$, we can decide whether $\mathcal{H}_1 \subseteq \mathcal{H}_2$ by checking if $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and $\mathcal{P}_1 \subseteq \mathcal{P}_2$. Also we can decide if $\mathcal{H}_1 = \mathcal{H}_2$ by checking if $\mathcal{L}_1 = \mathcal{L}_2$ and $\mathcal{P}_1 = \mathcal{P}_2$.

Suppose that $\mathcal{H}_1$ and $\mathcal{H}_2$ are relational grid-polyhedra, the case where $\mathcal{H}_1$ and $\mathcal{H}_2$ are non-relational grid-polyhedra is considered in Section 7.5. As we do not have an efficient algorithm for producing a reduced product grid-polyhedron, we will not always have the polyhedra represented in their most reduced form with respect to the grid points and therefore it is possible that a result of "don't know" will have to be returned in the case where the result should be $\mathcal{H}_1 \subseteq \mathcal{H}_2$, or $\mathcal{H}_1 = \mathcal{H}_2$. Example 6.29 will highlight this.

Example 6.29 *Consider $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$. Let*

$$\mathcal{P}_1 := \text{gen}\left(\varnothing, \varnothing, \begin{pmatrix} 1 & 4 & 6 & 6 & 5 \\ 3 & 0 & 3 & 5 & 6 \end{pmatrix}\right),$$

$$\mathcal{P}_2 := \text{con}\big(\{\ 1 \leq y,\ 3 \leq x + y,\ 0 \leq 2x - y \leq 9,\ 2x + y \leq 15\ -10 \leq x - 3y\}\big)$$

and

$$\mathcal{L}_1 = \mathcal{L}_2 = \text{gcon}\big(\{x \equiv_1 0, -x + y \equiv_3 0\}\big) = \text{ggen}\left(\varnothing, \varnothing, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 3 \end{pmatrix}\right).$$

Then we can see from Figure 6.9(a) that $\mathcal{H}_1 = \mathcal{H}_2$ and $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are weakly tight products. Since $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are not reduced products the comparison and equality test will return the result "don't know". Now consider $\mathcal{H}'_1 = (\mathcal{L}'_1, \mathcal{P}'_1)$ and $\mathcal{H}'_2 = (\mathcal{L}'_2, \mathcal{P}'_2)$ in $\mathbb{GP}_2$.

Let

$$\mathcal{P}'_1 := \mathrm{gen}\left(\varnothing, \varnothing, \begin{pmatrix} 2 & 6 & 6 & 4 & 2 \\ 1 & 1 & 4 & 6 & 5 \end{pmatrix}\right),$$

$$\mathcal{P}'_2 := \mathrm{con}\big(\{y \le 5,\ 2x + y \le 15,\ -1 \le 2x - y \le 9,\ 6 \le x + 2y\}\big)$$

and $\mathcal{L}'_1 = \mathcal{L}'_2 = \mathcal{L}_1 = \mathcal{L}_2$. *Then we can see from Figure 6.9(b) that* $\mathcal{H}'_1 = \mathcal{H}'_2$ *and* $(\mathcal{L}'_1, \mathcal{P}'_1)$ *and* $(\mathcal{L}'_2, \mathcal{P}'_2)$ *are tight products. Unfortunately the comparison and equality test will give the result "don't know".*

Let us now consider the complexities of both the comparison and equality operations. Let $\mathcal{L}_1 = \mathrm{ggen}(L, Q, P)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ where $m_1 = \#L + \#Q + \#P$ and $m_2 = \#\mathcal{C}_\mathcal{L}$. Let $\mathcal{P}_1 = \mathrm{gen}(L, R, P)$, $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_\mathcal{P})$ where $m_3 = \#L + \#R + \#P$ and $m_4 = \#\mathcal{C}_\mathcal{P}$. Assuming that the systems $\mathcal{G}_\mathcal{L}, \mathcal{G}_\mathcal{P}, \mathcal{C}_\mathcal{L}$ and $\mathcal{C}_\mathcal{P}$ are already available, the worst-case complexity of a comparison algorithm is $\mathrm{O}\big(n \max\{m_1 m_2, m_3 m_4\}\big)$. Note that, if $n \le \min\{m_1, m_2\}$, then it would be more efficient to compute the minimal forms for $\mathcal{C}_\mathcal{L}$ and $\mathcal{G}_\mathcal{L}$ before actually checking for comparison, hence obtaining the worst-case complexity $\mathrm{O}\big(n \max\{nm_1, nm_2, m_3 m_4\}\big)$; clearly, $\mathrm{O}\big(n \max\{n^2, m_3 m_4\}\big)$ is obtained if the two grid descriptions were already available in minimal form. Given that it is known that one grid is a subset of another and that one polyhedron is a subset of another, there are quicker tests for checking equality. The complexity of checking if $\mathcal{H}_1 = \mathcal{H}_2$ is just $\mathrm{O}\big(\max\{n^2, m_3^2, m_4^2\}\big)$.

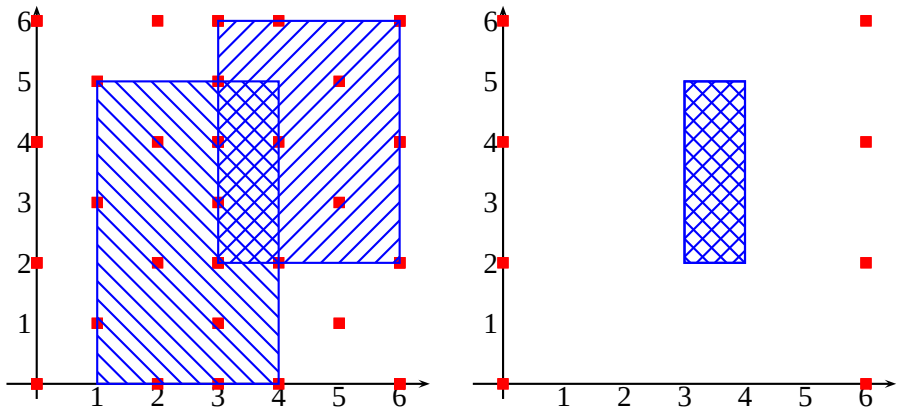
6.5.2 Intersection

For grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2) \in \mathbb{GP}_n$, then the *intersection* of $\mathcal{H}_1$ and $\mathcal{H}_2$, is defined as the pair $(\mathcal{L}_1 \cap \mathcal{L}_2, \mathcal{P}_1 \cap \mathcal{P}_2)$, which is the largest grid-polyhedron included in both $\mathcal{H}_1$ and $\mathcal{H}_2$. Then in theoretical terms, the intersection operation is the binary *meet* operator on the lattice $\mathbb{GP}_n$. It can easily be computed; if $\mathcal{H}_1 = \big(\mathrm{gcon}(\mathcal{C}_{\mathcal{L}1}), \mathrm{con}(\mathcal{C}_{\mathcal{P}1})\big)$ and $\mathcal{H}_2 = \big(\mathrm{gcon}(\mathcal{C}_{\mathcal{L}2}), \mathrm{con}(\mathcal{C}_{\mathcal{P}2})\big)$, then $\mathcal{H}_1 \cap \mathcal{H}_2 = \big(\mathrm{gcon}(\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}), \mathrm{con}(\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2})\big)$. However this operation of intersection does not preserve the given reduction of the polyhedron constraint systems.

Example 6.30 *Consider the grid-polyhedra* $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ *in* $\mathbb{GP}_2$ *where* $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}1})$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}2})$, $\mathcal{P}_1 = \mathrm{con}(\mathcal{C}_{\mathcal{P}1})$, $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_{\mathcal{P}2})$,

$$\begin{aligned} \mathcal{C}_{\mathcal{L}1} &:= \{x \equiv_1 0,\ x + y \equiv_2 0\} \quad \textit{and} \quad \mathcal{C}_{\mathcal{L}2} := \{x \equiv_3 0,\ y \equiv_2 0\}, \\ \mathcal{C}_{\mathcal{P}1} &:= \{1 \le x \le 4,\ 0 \le y \le 5\} \quad \textit{and} \quad \mathcal{C}_{\mathcal{P}2} := \{3 \le x \le 6,\ 2 \le y \le 6\}. \end{aligned}$$

The grids $\mathcal{L}_1$ *and* $\mathcal{L}_2$ *are illustrated by the filled squares in Figure 6.10(a). Then the grid intersection is* $\mathcal{L}_1 \cap \mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2})$*; thus, as* $\mathcal{C}_\mathcal{L} = \{x \equiv_6 0,\ y \equiv_2 0\}$ *is the minimal form of* $\mathcal{C}_{\mathcal{L}1} \cup \mathcal{C}_{\mathcal{L}2}$*, we have* $\mathcal{L}_1 \cap \mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$. *The grid* $\mathcal{L}_1 \cap \mathcal{L}_2$ *is illustrated by the filled squares in Figure 6.10(b).*



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \cap \mathcal{H}_2$.

Figure 6.10: Grid-Polyhedron intersection does not preserve the given reduction.

The polyhedron intersection is $\mathcal{P}_1 \cap \mathcal{P}_2 = \mathrm{con}(\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2})$*; thus, as* $\mathcal{C}_{\mathcal{P}} = \{3 \leq x \leq 4,\ 2 \leq y \leq 5\}$ *is a minimised form of* $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2}$*, we have* $\mathcal{P}_1 \cap \mathcal{P}_2 = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$.

Therefore it can be seen from Figure 6.10(b) that $\mathcal{H}_1 \cap \mathcal{H}_2 = \big(\mathrm{gcon}(\mathcal{C}_{\mathcal{L}}), \mathrm{con}(\mathcal{C}_{\mathcal{P}})\big)$*, where*

$$\mathcal{C}_{\mathcal{L}} := \{x \equiv_6 0,\ y \equiv_2 0\} \quad \textit{and} \quad \mathcal{C}_{\mathcal{P}} := \{3 \leq x \leq 4,\ 2 \leq y \leq 5\},$$

is empty, hence this also shows why we require an operation to detect emptiness. Also note that $(\mathcal{L}_1, \mathcal{P}_1)$ *and* $(\mathcal{L}_2, \mathcal{P}_2)$ *are tight products. We can see that* $\mathcal{H}_1 \cap \mathcal{H}_2 = \big(\mathrm{gcon}(\mathcal{C}_{\mathcal{L}}), \mathrm{con}(\mathcal{C}_{\mathcal{P}})\big)$ *is empty and* $\mathcal{C}_{\mathcal{P}}$ *is not even a weakly tight polyhedron constraint system for* $\mathcal{H}_1 \cap \mathcal{H}_2$*. Note that even if* $(\mathcal{L}_1, \mathcal{P}_1)$ *were a reduced product, the resulting grid-polyhedron pair after the intersection would also not have been at least weakly tight.*

Example 6.31 shows that if we have two grid-polyhedra, $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1), \mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ where $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are both reduced products, then after the intersection is performed, the resulting grid-polyhedron is $\mathcal{H}_1 \cap \mathcal{H}_2 = (\mathcal{L}, \mathcal{P})$ and $(\mathcal{L}, \mathcal{P})$ is not a reduced product.

Example 6.31 *Consider* $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ *in* $\mathbb{GP}_2$*. Let*

$$\mathcal{P}_1 := \mathrm{con}\big(\{1 \leq x \leq 4, 1 \leq y \leq 4\}\big), \qquad \mathcal{P}_2 := \mathrm{con}\big(\{2 \leq x \leq 5, 2 \leq y \leq 5\}\big)$$

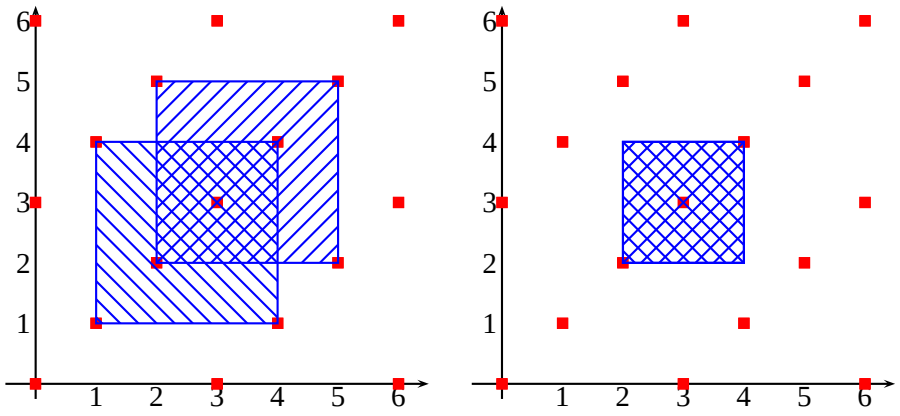
and

$$\mathcal{L}_1 = \mathcal{L}_2 = \mathrm{gcon}\big(\{x \equiv_1 0, -x + y \equiv_3 0\}\big).$$

The grid-polyhedra $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are illustrated in Figure 6.11(a). Then the grid-polyhedron intersection* $\mathcal{H}_1 \cap \mathcal{H}_2 = (\mathcal{L}_1 \cap \mathcal{L}_2, \mathcal{P}_1 \cap \mathcal{P}_2)$ *is given by*

$$\Big(\mathrm{gcon}\big(\{x \equiv_1 0, -x + y \equiv_3 0\}\big), \mathrm{con}\big(\{2 \leq x \leq 4, 2 \leq y \leq 4\}\big)\Big).$$

The grid-polyhedron $\mathcal{H}_1 \cap \mathcal{H}_2$ *is illustrated in Figure 6.11(b) and it can be seen that* $(\mathcal{L}_1 \cap \mathcal{L}_2, \mathcal{P}_1 \cap$



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \cap \mathcal{H}_2$.

Figure 6.11: Grid-Polyhedron intersection.

$\mathcal{P}_2$) *is not a reduced product.*

The only partially reduced products the operation of intersection will preserve are the smash and constraint products.

6.5.3 Join

For grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2) \in \mathbb{GP}_n$, the *join* of $\mathcal{H}_1$ and $\mathcal{H}_2$, is defined as the pair $(\mathcal{L}_1 \oplus \mathcal{L}_2, \mathcal{P}_1 \oplus \mathcal{P}_2)$, which is the smallest grid-polyhedron containing both $\mathcal{H}_1$ and $\mathcal{H}_2$. Then in theoretical terms, the join operation is the binary *join* operators on the lattice $\mathbb{GP}_n$. It can easily be computed; if $\mathcal{H}_1 = \big(\mathrm{ggen}(\mathcal{G}_{\mathcal{L}1}), \mathrm{gen}(\mathcal{G}_{\mathcal{P}1})\big)$ and $\mathcal{H}_2 = \big(\mathrm{ggen}(\mathcal{G}_{\mathcal{L}2}), \mathrm{gen}(\mathcal{G}_{\mathcal{P}2})\big)$, then $\mathcal{H}_1 \oplus \mathcal{H}_2 = \big(\mathrm{ggen}(\mathcal{G}_{\mathcal{L}1} \cup \mathcal{G}_{\mathcal{L}2}), \mathrm{gen}(\mathcal{G}_{\mathcal{P}1} \cup \mathcal{G}_{\mathcal{P}2})\big)$. Unlike the operation of intersection the operation of join does respect if the grid-polyhedra are reduced products. Let $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$, then if $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are reduced products then every generating point of the polyhedra is also a grid point. Therefore after the operation of join is performed every generating point will still be a grid point. Example 6.32 demonstrates this point.

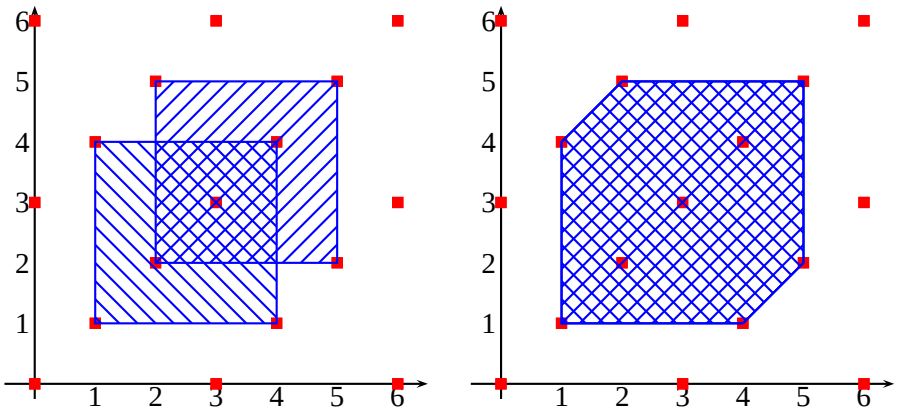
Example 6.32 *Consider $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$. Let $\mathcal{P}_1 = \mathrm{gen}(\varnothing, \varnothing, P_1)$ and $\mathcal{P}_2 = \mathrm{gen}(\varnothing, \varnothing, P_2)$ in $\mathbb{CP}_2$, where*

$$P_1 := \begin{pmatrix} 1 & 1 & 4 & 4 \\ 1 & 4 & 1 & 4 \end{pmatrix}, \qquad P_2 := \begin{pmatrix} 2 & 2 & 5 & 5 \\ 2 & 5 & 2 & 5 \end{pmatrix}$$

and $\mathcal{L}_1 = \mathcal{L}_2 = \mathrm{ggen}(\varnothing, \varnothing, P_0)$ in $\mathbb{G}_2$, where

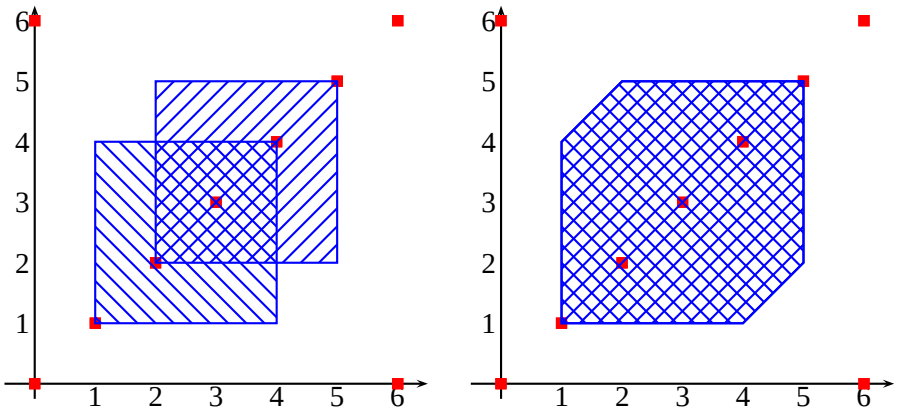
$$P_0 := \begin{pmatrix} 1 & 0 & 0 \\ 1 & 3 & 0 \end{pmatrix}.$$

The grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$ are illustrated in Figure 6.12(a). The grid-polyhedron join $\mathcal{H}_1 \oplus$



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$.

Figure 6.12: Grid-Polyhedron join.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$.

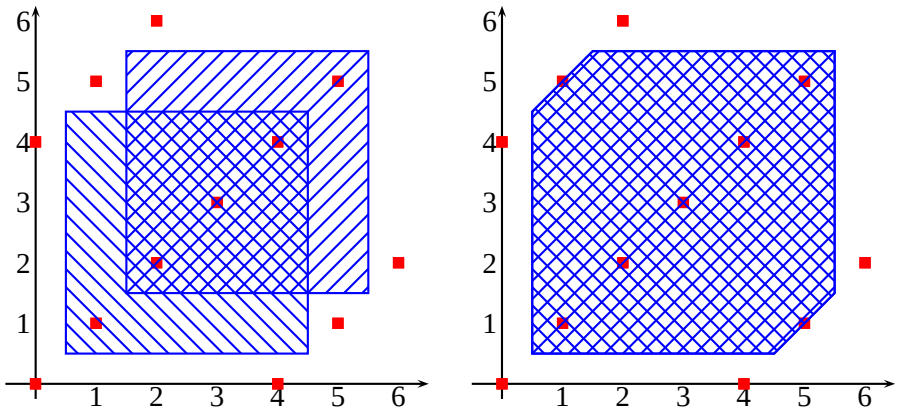
Figure 6.13: Grid-Polyhedron join does not respect tight products.

$\mathcal{H}_2 = (\mathcal{L}_1 \oplus \mathcal{L}_2, \mathcal{P}_1 \oplus \mathcal{P}_2)$ is given by $\big(\text{ggen}(\varnothing, \varnothing, P_0 \cup P_0), \text{gen}(\varnothing, \varnothing, P_1 \cup P_2)\big)$, since $\mathcal{L}_1 \oplus \mathcal{L}_2$ takes the union of the grids generator systems and $\mathcal{P}_1 \oplus \mathcal{P}_2$ takes the union of the polyhedra generator systems. Thus, the generator system of the polyhedron is given by

$$\left(\varnothing, \varnothing, \begin{pmatrix} 1 & 1 & 4 & 2 & 5 & 5 \\ 1 & 4 & 1 & 5 & 5 & 2 \end{pmatrix}\right).$$

The grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$ is illustrated in Figure 6.12(b). It can be seen that the grid-polyhedron pair $(\mathcal{L}_1 \oplus \mathcal{L}_2, \mathcal{P}_1 \oplus \mathcal{P}_2)$ is a reduced product as every vertex of $\mathcal{P}_1 \oplus \mathcal{P}_2$ is a point of the grid $\mathcal{L}_1 \oplus \mathcal{L}_2$.

However the operation of join does not respect if the grid-polyhedra are tight or weakly tight products. Example 6.33 establishes this point.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$.

Figure 6.14: Grid-Polyhedron join requires the grid-polyhedra pairs to be weakly tight products.

Example 6.33 *Consider* $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ *in* $\mathbb{GP}_2$. *Let* $\mathcal{P}_1 = \text{gen}(\varnothing, \varnothing, P_1)$ *and* $\mathcal{P}_2 = \text{gen}(\varnothing, \varnothing, P_2)$ *in* $\mathbb{CP}_2$, *where*

$$P_1 := \begin{pmatrix} 1 & 1 & 4 & 4 \\ 1 & 4 & 1 & 4 \end{pmatrix}, \qquad P_2 := \begin{pmatrix} 2 & 2 & 5 & 5 \\ 2 & 5 & 2 & 5 \end{pmatrix}$$

and $\mathcal{L}_1 = \mathcal{L}_2 = \text{ggen}(\varnothing, \varnothing, P_0)$ *in* $\mathbb{G}_2$, *where*

$$P_0 := \begin{pmatrix} 1 & 0 & 0 \\ 1 & 6 & 0 \end{pmatrix}.$$

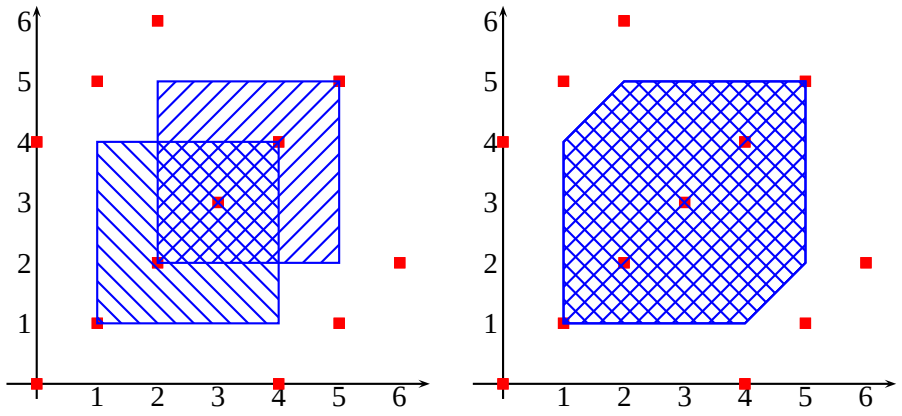
The grid-polyhedra $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are illustrated in Figure 6.13(a). The grid-polyhedron join* $\mathcal{H}_1 \oplus \mathcal{H}_2 = (\mathcal{L}_1 \oplus \mathcal{L}_2, \mathcal{P}_1 \oplus \mathcal{P}_2)$ *is given by* $\big(\text{ggen}(\varnothing, \varnothing, P_0 \cup P_0), \text{gen}(\varnothing, \varnothing, P_1 \cup P_2)\big)$; *thus, the generator system of the polyhedron is given by*

$$\left(\varnothing, \varnothing, \begin{pmatrix} 1 & 1 & 4 & 2 & 5 & 5 \\ 1 & 4 & 1 & 5 & 5 & 2 \end{pmatrix}\right).$$

The grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$ *is illustrated in Figure 6.13(b). It can be seen that the grid-polyhedron pair* $(\mathcal{L}_1 \oplus \mathcal{L}_2, \mathcal{P}_1 \oplus \mathcal{P}_2)$ *is not a tight or weakly tight product as the constraint* $x - y \leq 3$ *is not saturated by a grid point.*

The partially reduced products the operation of join will preserve are the smash, constraint and reduced products.

Given two grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$, Example 6.34 shows that if $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are not weakly tight products then the resulting grid-polyhedron, after the operation of join is performed, can have more points compared to the result if $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are weakly tight products.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$.

Figure 6.15: Grid-Polyhedron join requires the grid-polyhedra pairs to be weakly tight products.

Example 6.34 *Consider* $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ *in* $\mathbb{GP}_2$. *Let* $\mathcal{P}_1 = \operatorname{gen}(\varnothing, \varnothing, P_1)$ *and* $\mathcal{P}_2 = \operatorname{gen}(\varnothing, \varnothing, P_2)$ *in* $\mathbb{CP}_2$, *where*

$$P_1 := \begin{pmatrix} 0.5 & 0.5 & 4.5 & 4.5 \\ 0.5 & 4.5 & 0.5 & 4.5 \end{pmatrix}, \qquad P_2 := \begin{pmatrix} 1.5 & 1.5 & 5.5 & 5.5 \\ 1.5 & 5.5 & 1.5 & 5.5 \end{pmatrix}$$

and $\mathcal{L}_1 = \mathcal{L}_2 = \operatorname{ggen}(\varnothing, \varnothing, P_0)$ *in* $\mathbb{G}_2$, *where*

$$P_0 := \begin{pmatrix} 1 & 0 & 0 \\ 1 & 4 & 0 \end{pmatrix}.$$

The grid-polyhedra $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are illustrated in Figure 6.14(a). The grid-polyhedron join* $\mathcal{H}_1 \oplus \mathcal{H}_2 = (\mathcal{L}_1 \oplus \mathcal{L}_2, \mathcal{P}_1 \oplus \mathcal{P}_2)$ *is given by* $\big(\operatorname{ggen}(\varnothing, \varnothing, P_0 \cup P_0), \operatorname{gen}(\varnothing, \varnothing, P_1 \cup P_2)\big)$; *thus, the generator system of the polyhedron is given by*

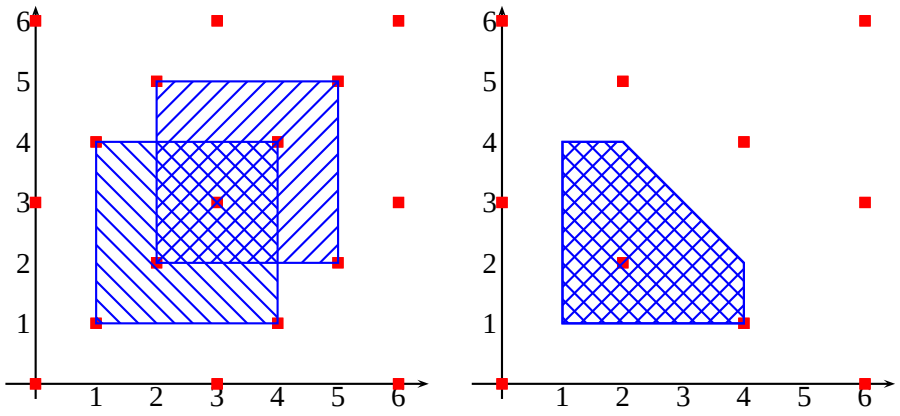
$$\left(\varnothing, \varnothing, \begin{pmatrix} 0.5 & 0.5 & 4.5 & 1.5 & 5.5 & 5.5 \\ 0.5 & 4.5 & 0.5 & 5.5 & 5.5 & 1.5 \end{pmatrix}\right).$$

The grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$ *is illustrated in Figure 6.14(b) and it can be seen that* $\mathcal{H}_1 \oplus \mathcal{H}_2$ *contains the points* $(1, 5)^{\mathrm{T}}$ *and* $(5, 1)^{\mathrm{T}}$.

Now consider $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}'_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}'_2)$ *in* $\mathbb{GP}_2$. *Let* $\mathcal{P}'_1 = \operatorname{gen}(\varnothing, \varnothing, P'_1)$ *and* $\mathcal{P}'_2 = \operatorname{gen}(\varnothing, \varnothing, P'_2)$ *in* $\mathbb{CP}_2$, *where*

$$P_1 := \begin{pmatrix} 1 & 1 & 4 & 4 \\ 1 & 4 & 1 & 4 \end{pmatrix}, \quad \textit{and} \quad P_2 := \begin{pmatrix} 2 & 2 & 5 & 5 \\ 2 & 5 & 2 & 5 \end{pmatrix}.$$

The grid-polyhedra $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are also illustrated in Figure 6.15(a) and it can be seen that* $(\mathcal{L}_1, \mathcal{P}'_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}'_2)$ *are weakly tight. The grid-polyhedron join* $\mathcal{H}_1 \oplus \mathcal{H}_2 = (\mathcal{L}_1 \oplus$



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \ominus \mathcal{H}_2$.

Figure 6.16: Grid-Polyhedron difference.

$\mathcal{L}_2, \mathcal{P}'_1 \oplus \mathcal{P}'_2$) is given by $\big(\mathrm{ggen}(\varnothing, \varnothing, P_0 \cup P_0), \mathrm{gen}(\varnothing, \varnothing, P'_1 \cup P'_2)\big)$*; thus, the generator system of the polyhedron is given by*

$$\left(\varnothing, \varnothing, \begin{pmatrix} 1 & 1 & 4 & 2 & 5 & 5 \\ 1 & 4 & 1 & 5 & 5 & 2 \end{pmatrix}\right).$$

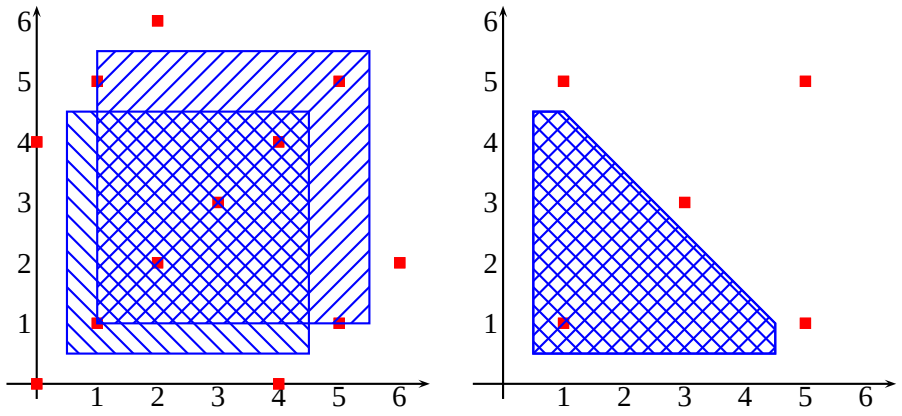
The grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$ is illustrated in Figure 6.15(b) and it can be seen that $\mathcal{H}_1 \oplus \mathcal{H}_2$ does not contains the points $(1, 5)^{\mathrm{T}}$ and $(5, 1)^{\mathrm{T}}$.

6.5.4 Difference

Let us recall from Section 4.5 that the grid difference of $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \ominus \mathcal{L}_2$, is defined as the smallest grid containing the set-theoretic difference of $\mathcal{L}_1$ and $\mathcal{L}_2$. Also the convex polyhedral difference (or poly-difference) of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted by $\mathcal{P}_1 \ominus \mathcal{P}_2$, is defined as the smallest convex polyhedron containing the set-theoretic difference of $\mathcal{P}_1$ and $\mathcal{P}_2$. Therefore for any pair of grid-polyhedra $\mathcal{H}_1, \mathcal{H}_2 \in \mathbb{GP}_n$, the *grid-polyhedron difference* of $\mathcal{H}_1$ and $\mathcal{H}_2$, denoted by $\mathcal{H}_1 \ominus \mathcal{H}_2$, is defined as the smallest grid-polyhedron containing the set-theoretic difference of $\mathcal{H}_1$ and $\mathcal{H}_2$. The grid-polyhedron difference is computed by taking the difference of each of the components of the product, specifically

$$\mathcal{H}_1 \ominus \mathcal{H}_2 := \big(\mathcal{L}_1 \ominus \mathcal{L}_2, \mathcal{P}_1 \ominus \mathcal{P}_2\big).$$

Example 6.35 shows that if we have two grid-polyhedra, $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1), \mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ where $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are both reduced products, then after the grid-polyhedron difference is performed the resulting grid-polyhedra pair is not necessarily a reduced product.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \ominus \mathcal{H}_2$.

Figure 6.17: Grid-Polyhedron difference requires the grid-polyhedra pairs to be weakly tight products.

Example 6.35 *Consider* $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ *and* $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ *in* $\mathbb{GP}_2$. *Let*

$$\mathcal{P}_1 := \mathrm{con}\big(\{1 \le x \le 4, 1 \le y \le 4\}\big), \qquad \mathcal{P}_2 := \mathrm{con}\big(\{2 \le x \le 5, 2 \le y \le 5\}\big)$$

and

$$\mathcal{L}_1 := \mathrm{gcon}\big(\{x \equiv_1 0, -x + y \equiv_3 0\}\big), \qquad \mathcal{L}_2 := \mathrm{gcon}\big(\{x \equiv_2 1, x + 2y \equiv_6 3\}\big).$$

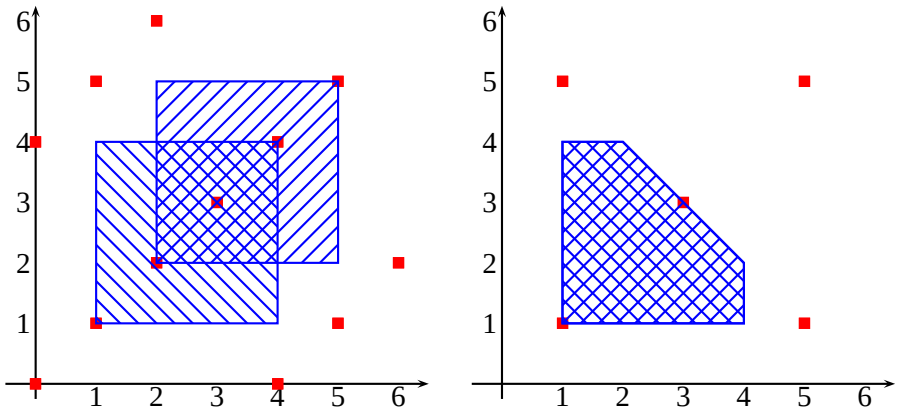
The grid-polyhedra $\mathcal{H}_1$ *and* $\mathcal{H}_2$ *are illustrated in Figure 6.16(a). Then the grid-polyhedron difference* $\mathcal{H}_1 \ominus \mathcal{H}_2 = (\mathcal{L}_1 \ominus \mathcal{L}_2, \mathcal{P}_1 \ominus \mathcal{P}_2)$ *is given by*

$$\Big(\mathrm{gcon}\big(\{x \equiv_2 0, x + 2y \equiv_6 0\}\big), \mathrm{con}\big(\{1 \le x \le 4, 1 \le y \le 4, x + y \le 6\}\big)\Big).$$

The grid-polyhedron $\mathcal{H}_1 \ominus \mathcal{H}_2$ *is illustrated in Figure 6.16(b) and it can be seen that* $(\mathcal{L}_1 \ominus \mathcal{L}_2, \mathcal{P}_1 \ominus \mathcal{P}_2)$ *is not a reduced product.*

Like the grid-polyhedron intersection operation, the grid-polyhedron difference operation does not respect if the grid-polyhedra are tight or weakly tight products. So after the operation is performed the resulting grid-polyhedron may no longer be a tight or weakly tight product. The only partially reduced products the operation of difference will preserve are the smash and constraint products.

Given two grid-polyhedra $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$, Example 6.36 shows that if $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are not weakly tight products then the resulting grid-polyhedron, after the operation of difference is performed, can have less points compared to the result when $(\mathcal{L}_1, \mathcal{P}_1)$ and $(\mathcal{L}_2, \mathcal{P}_2)$ are weakly tight products.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$. (b) Grid-polyhedron $\mathcal{H}_1 \oplus \mathcal{H}_2$.

Figure 6.18: Grid-Polyhedron difference requires the grid-polyhedra pairs to be weakly tight products.

Example 6.36 *Consider $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$. Let*

$$\mathcal{P}_1 := \mathrm{con}\big(\{0.5 \leq x \leq 4.5, 0.5 \leq y \leq 4.5\}\big), \qquad \mathcal{P}_2 := \mathrm{con}\big(\{1 \leq x \leq 5.5, 1 \leq y \leq 5.5\}\big)$$

and

$$\mathcal{L}_1 := \mathrm{gcon}\big(\{x \equiv_1 0, -x + y \equiv_4 0\}\big), \qquad \mathcal{L}_2 := \mathrm{gcon}\big(\{x \equiv_2 0, x + y \equiv_4 0\}\big).$$

The grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$ are illustrated in Figure 6.17(a). Then the grid-polyhedron difference $\mathcal{H}_1 \ominus \mathcal{H}_2 = (\mathcal{L}_1 \ominus \mathcal{L}_2, \mathcal{P}_1 \ominus \mathcal{P}_2)$ is given by

$$\Big(\mathrm{gcon}\big(\{x \equiv_2 1, x + y \equiv_4 2\}\big), \mathrm{con}\big(\{0.5 \leq x \leq 4.5, 0.5 \leq y \leq 4.5, x + y \leq 5.5\}\big)\Big).$$

The grid-polyhedron $\mathcal{H}_1 \ominus \mathcal{H}_2$ is illustrated in Figure 6.17(b) and it can be seen that $\mathcal{H}_1 \ominus \mathcal{H}_2$ only contains the point $(1, 1)^{\mathrm{T}}$.

Now consider $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}'_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}'_2)$ in $\mathbb{GP}_2$. Let

$$\mathcal{P}_1 := \mathrm{con}\big(\{1 \leq x \leq 4, 1 \leq y \leq 4\}\big), \qquad \mathcal{P}_2 := \mathrm{con}\big(\{2 \leq x \leq 5, 2 \leq y \leq 5\}\big).$$

The grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$ are also illustrated in Figure 6.18(a) and it can be seen that $(\mathcal{L}_1, \mathcal{P}'_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}'_2)$ are weakly tight. Then the grid-polyhedron difference $\mathcal{H}_1 \ominus \mathcal{H}_2 = (\mathcal{L}_1 \ominus \mathcal{L}_2, \mathcal{P}_1 \ominus \mathcal{P}_2)$ is given by

$$\Big(\mathrm{gcon}\big(\{x \equiv_2 1, x + y \equiv_4 2\}\big), \mathrm{con}\big(\{1 \leq x \leq 4, 1 \leq y \leq 4, x + y \leq 6\}\big)\Big).$$

The grid-polyhedron $\mathcal{H}_1 \ominus \mathcal{H}_2$ is illustrated in Figure 6.18(b) and it can be seen that $\mathcal{H}_1 \ominus \mathcal{H}_2$ also contains the point $(3, 3)^{\mathrm{T}}$.

6.5.5 Affine Image and Pre-image

Affine transformations for the vector space $\mathbb{R}^n$ will map hyperplanes to hyperplanes, preserve intersection properties between hyperplanes and preserve ratios of distances between points along a hyperplane; such transformations can be represented by matrices in $\mathbb{R}^{n\times n}$. It follows that the set $\mathbb{GP}_n$ is closed under the set of all affine transformations for $\mathbb{R}^n$. Simple and useful linear affine transformations for numerical domains, including the grid-polyhedra, are provided by the 'single update' affine image and affine pre-image operators.

Given a grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$, a variable x_k and linear expression $e = \langle \mathbf{a}, \mathbf{x} \rangle + b$ with coefficients in $\mathbb{Q}$, the *affine image operator* $\phi(\mathcal{H}, x_k, e)$ maps the grid-polyhedron $\mathcal{H}$ to

$$\big(\phi(\mathcal{L}, x_k, e), \phi(\mathcal{P}, x_k, e)\big).$$

Conversely, the *affine pre-image operator* $\phi^{-1}(\mathcal{H}, x_k, e)$ maps the grid-polyhedron $\mathcal{H}$ to

$$\big(\phi^{-1}(\mathcal{L}, x_k, e), \phi^{-1}(\mathcal{P}, x_k, e)\big).$$

Observe that the affine image $\phi(\mathcal{H}, x_k, e)$ and pre-image $\phi^{-1}(\mathcal{H}, x_k, e)$ are invertible if and only if the coefficient a_k in the vector $\mathbf{a}$ is non-zero. Note that as the affine image and pre-image operations preserve intersection properties between hyperplanes and preserve ratios of distances between points along a hyperplane we have that if $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ and $(\mathcal{L}, \mathcal{P})$ is a reduced product (resp. constraint, weakly tight or tight product), then after the affine image (resp. pre-image) operation is performed the resulting grid-polyhedron pair $\big(\phi(\mathcal{L}, x_k, e), \phi(\mathcal{P}, x_k, e)\big)$ (resp. $\big(\phi^{-1}(\mathcal{L}, x_k, e), \phi^{-1}(\mathcal{P}, x_k, e)\big)$) is also a reduced product (resp. constraint, weakly tight or tight product).

The *generalized affine image* (resp., *generalized affine pre-image*) is an extension of the affine image (resp., affine pre-image) operator defined above. Given a grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$, linear expressions $e' = \langle \mathbf{c}, \mathbf{x} \rangle + d$ and $e = \langle \mathbf{a}, \mathbf{x} \rangle + b$ with coefficients in $\mathbb{Q}$, $f \in \mathbb{Q}$ and $\bowtie \in \{\leq, =, \geq\}$, the generalized affine image operator $\psi = \psi(\mathcal{H}, e', e, f, \bowtie)$ is defined as

$$\big(\psi(\mathcal{L}, e', e, f), \psi(\mathcal{P}, e', e, \bowtie)\big).$$

where $\psi(\mathcal{L}, e', e, f)$ is defined as

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^n : (\mathbf{v}, \mathbf{w}) \in \psi \iff \big(\langle \mathbf{c}, \mathbf{w} \rangle + d \equiv_f \langle \mathbf{a}, \mathbf{v} \rangle + b\big) \wedge \Big(\bigwedge_{\substack{0 \leq i < n \\ c_i = 0}} w_i = v_i \Big)$$

and where $\psi(\mathcal{P}, e', e, \bowtie)$ is defined as

$$\forall \mathbf{v}, \mathbf{w} \in \mathbb{R}^n : (\mathbf{v}, \mathbf{w}) \in \psi \iff \big(\langle \mathbf{c}, \mathbf{w} \rangle + d \bowtie \langle \mathbf{a}, \mathbf{v} \rangle + b\big) \wedge \Big(\bigwedge_{\substack{0 \leq i < n \\ c_i = 0}} w_i = v_i \Big).$$

Note that, when $e' = x_k$ and $f = 0$, then the transformation is equivalent to the standard affine transformation on $\mathcal{L}$ with respect to the variable x_k and the affine expression e; that is

$$\psi(\mathcal{L}, x_k, e, 0) = \phi(\mathcal{L}, x_k, e).$$

Also note that, when $e' = x_k$ and $\bowtie \in \{=\}$, then the transformation is equivalent to the standard affine transformation on $\mathcal{P}$ with respect to the variable x_k and the affine expression e; that is

$$\psi(\mathcal{P}, x_k, e, =) = \phi(\mathcal{P}, x_k, e).$$

6.5.6 Widening

Recall from Chapter 5 that we have described two widening operators for the grid domain and also note that there are several possible widenings for the polyhedron domain [4, 6, 43, 44]. Therefore let $\nabla_{\mathcal{L}}$ be a widening on the grid domain and $\nabla_{\mathcal{P}}$ a widening for the polyhedron domain. Let $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$, then

$$\mathcal{H}_1 \nabla \mathcal{H}_2 := \big(\mathcal{L}_1 \nabla_{\mathcal{L}} \mathcal{L}_2, \mathcal{P}_1 \nabla_{\mathcal{P}} \mathcal{P}_2\big).$$

As noted in Section 6.2 if we use the standard widening for polyhedra, then ∇ defined as above is a widening for the all partially reduced product domains defined here except for the reduced product domain. This is due to the fact that for a polyhedron widening to satisfy the ascending chain condition, at each stage the constraint representation must decrease by at least one element. However if the reduction method for the grid-polyhedra were to include the adding of constraints to the representation, which could occur with the a reduced product, this decrease will not be achieved.

The widening function for grid-polyhedra will preserve the given reduction. Example 6.37 shows this for the case where both grid-polyhedra pairs are reduced products.

Example 6.37 *Consider $\mathcal{H}_1 = (\mathcal{L}_1, \mathcal{P}_1)$ and $\mathcal{H}_2 = (\mathcal{L}_2, \mathcal{P}_2)$ in $\mathbb{GP}_2$. Let*

$$\mathcal{P}_1 := \mathrm{con}\big(\{0 \le x,\ 0 \le y,\ -2 \le x - y \le 2,\ x + y \le 6\}\big),$$
$$\mathcal{P}_2 := \mathrm{con}\big(\{0 \le x \le 6,\ 0 \le y \le 6,\ x + y \le 8\}\big)$$

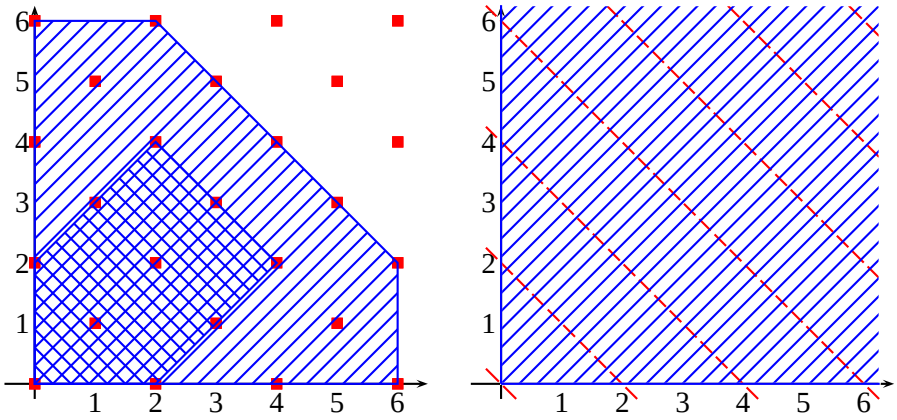
and

$$\mathcal{L}_1 := \mathrm{gcon}\big(\{x \equiv_2 0,\ y \equiv_2 0\}\big), \qquad \mathcal{L}_2 := \mathrm{gcon}\big(\{x \equiv_1 0,\ x + y \equiv_2 0\}\big).$$

The grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$ are illustrated in Figure 6.19(a). Then $\mathcal{L}_1 \nabla_{\mathcal{L}} \mathcal{L}_2 := \mathrm{gcon}\big(\{x + y \equiv_2 0\}\big)$ and

$$\mathcal{P}_1 \nabla_{\mathcal{P}} \mathcal{P}_2 := \mathrm{con}\big(\{0 \le x,\ 0 \le y\}\big).$$

The grid-polyhedron $\mathcal{H}_1 \nabla \mathcal{H}_2$ is illustrated in Figure 6.19(b) and it can be seen that the grid-polyhedron pair, $\big(\mathcal{L}_1 \nabla_{\mathcal{L}} \mathcal{L}_2, \mathcal{P}_1 \nabla_{\mathcal{P}} \mathcal{P}_2\big)$ is a reduced product.



(a) Grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$.

(b) Grid-polyhedron $\mathcal{H}_1 \nabla \mathcal{H}_2$.

Figure 6.19: Grid-Polyhedron Widening.

6.6 Discussion

As noted in Section 6.4 one way to improve the polyhedron constraint bounds would be to add new constraints to the polyhedron representation. The reason why this method was not chosen for our reduction was that standard polyhedron widening would not be guaranteed to terminate, see Section 6.5.6. Although the method of adding polyhedron constraints to minimise a grid-polyhedron to a reduced product will not work with a standard widening we will discuss below some of the methods that could be used if the widening is considered as an extrapolation operation instead.

6.6.1 Utilising Grid Congruences to Add Constraints

Consider the grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} := \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ and $\mathcal{C}_\mathcal{L}$ is in minimal form. Now for each proper congruence in the grid representation we can find the maximal and minimal values where the congruence would bound the polyhedron from above and below respectively.

Definition 6.38 (Grid Bounded Constraint System.) *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ be a grid-polyhedron and $(\mathcal{L}, \mathcal{P})$ be a weakly tight product, where $\mathcal{P} = \mathrm{con}(\mathcal{P})$, $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$ and $\mathcal{C}_\mathcal{L}$ is in minimal form. Then $\mathcal{C}_\mathcal{P}$ is a* grid bounded constraint system *for $\mathcal{H}$ if, for each proper congruence $\beta = \big(\langle \mathbf{v}, \mathbf{x} \rangle \equiv_f b\big) \in \mathcal{C}_\mathcal{L}$:*

1. *if $\mathcal{P}$ is bounded in the direction $\mathbf{v}$, then there exists $\nu = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d\big) \in \mathcal{C}_\mathcal{P}$.*
2. *there exists a point $\mathbf{p} \in \mathcal{P}$ such that $\langle \mathbf{v}, \mathbf{p} \rangle = d$;*
3. *if $\mathcal{P}$ is bounded in the direction $-\mathbf{v}$, then there exists $\nu' = \big(\langle -\mathbf{v}, \mathbf{x} \rangle \leq d'\big) \in \mathcal{C}_\mathcal{P}$.*
4. *there exists a point $\mathbf{p}' \in \mathcal{P}$ such that $\langle -\mathbf{v}, \mathbf{p}' \rangle = d'$;*

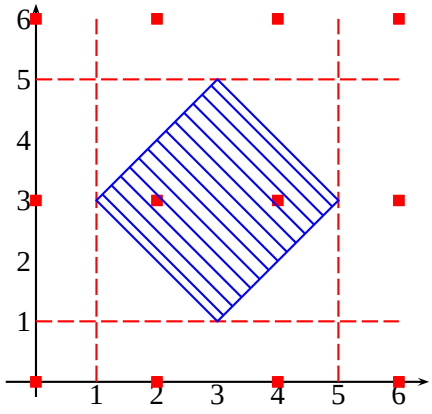


Figure 6.20: Grid Bounded Constraint System.

Example 6.39 *Consider the grid-polyhedron* $\mathcal{H} = (\mathcal{L}, \mathcal{P})$*, where* $\mathcal{L} = \text{gcon}(\mathcal{C}_\mathcal{L})$ *and* $\mathcal{C}_\mathcal{L} := \{x \equiv_2 0,\ y \equiv_3 0\}$*. Let* $\mathcal{P}$ *be the polyhedron given by the constraint system*

$$\mathcal{C}_\mathcal{P} := \big\{4 \leq x + y \leq 8, -2 \leq y - x \leq 2\big\}.$$

Then $\mathcal{H}$ *is a weakly tight product and can be seen in Figure 6.20. The grid bounded constraint system for* $\mathcal{H}$ *is given by*

$$\mathcal{C} = \text{con}\big(\{1 \leq x \leq 5, 1 \leq y \leq 5\}\big).$$

The constraints of $\mathcal{C}$ *are illustrated in Figure 6.20 by the dashed lines.*

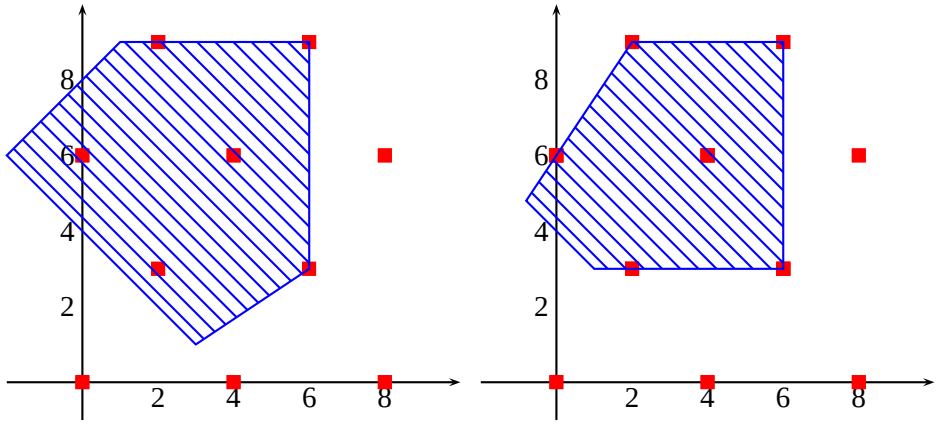
Now if necessary the classical approaches of branch and bound or the cutting plane method can be applied to the larger set of constraints, see Section 6.6.2. Note also that for each grid we can produce the smallest rectilinear grid that contains $\mathcal{L}$ by computing the covering box. Suppose therefore that we have a rectilinear grid $\mathcal{L}' = \text{gcon}(\mathcal{C}_\mathcal{L}')$, such that the congruences are given by $\mathcal{C}_\mathcal{L}' := \{x_i \equiv_f b_i\}$. Then we can also produce the rectilinear grid bounded constraints.

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ be a grid-polyhedron where $(\mathcal{L}, \mathcal{P})$ is a weakly tight and constraint product. Also let $\mathcal{L} = \text{gcon}(\mathcal{C}_\mathcal{L})$ be a relational grid, $\mathcal{P} = \text{con}(\mathcal{C}_\mathcal{P})$ and $\mathcal{C}_\mathcal{P}'$ be the set of rectilinear and relational grid bounded constraints for $\mathcal{H}$ which have been generated from the congruences systems for $\mathcal{L}$ and $\mathcal{L}'$, where $\mathcal{L}'$ is the smallest rectilinear grid containing $\mathcal{L}$. Let $\mathcal{C}_\mathcal{P}''$ be the congruence system returned by Algorithm 3 when applied to $\mathcal{H} = \big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P}')\big)$. Then $\big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{P}'')\big)$ is a weakly tight product and $\mathcal{H} = \big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{P}'')\big)$. Example 6.40 demonstrates this.

Example 6.40 *Consider the grid-polyhedron* $\mathcal{H} = (\mathcal{L}, \mathcal{P})$*, where* $\mathcal{L} = \text{gcon}(\mathcal{C}_\mathcal{L})$*,* $\mathcal{C}_\mathcal{L} := \{x \equiv_2 0,\ -3x + 2y \equiv_{12} 0\}$ *and* $\mathcal{P}$ *is given by the constraint system*

$$\mathcal{C}_\mathcal{P} := \{x \leq 6,\ y \leq 9,\ 4 \leq x + y,\ y - x \leq 8,\ 2x - 3y \leq 3\}.$$

$\mathcal{H}$ *can be seen in Figure 6.21(a). Then a rectilinear grid for* $\mathcal{L}$ *is* $\mathcal{L}'$ *where* $\mathcal{L}' = \text{gcon}(\mathcal{C}_\mathcal{L}')$ *and*



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{P}{}'')\big)$.

Figure 6.21: Adding constraints to a grid-polyhedron.

$\mathcal{C}_\mathcal{L}{}' := \{x \equiv_2 0,\ y \equiv_3 0\}$. *From* $\mathcal{L}$ *and* $\mathcal{L}'$ *we can calculate the relational and rectilinear grid bounded constraints given by*

$$\mathcal{C}_\mathcal{P}{}' := \{-2 \leq x \leq 6,\ 1 \leq y \leq 9,\ -12 \leq -3x + 2y \leq 18\}.$$

Now applying Algorithm 3 *to* $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}')\big)$ *we get the constraint system* $\mathcal{C}_\mathcal{P}{}''$ *and together with the constraint* $\mathcal{C}_\mathcal{P}$ *we get the new constraint system* $\mathrm{con}(\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{P}{}'')$ *where*

$$\mathcal{C}_\mathcal{P}{}'' := \{-2 \leq x \leq 6,\ 3 \leq y \leq 9,\ -12 \leq -3x + 2y \leq 12\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{P}{}'')\big)$ *can be seen in Figure 6.21(b). The constraint system* $\mathcal{C}_\mathcal{P} \cup \mathcal{C}_\mathcal{P}{}''$ *is weakly tight for the grid-polyhedron* $\mathcal{H}$.

For any grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ is relational and $\mathcal{C}_\mathcal{L}$ is in minimal form, the complexity of creating the $2n-2$ new relational and rectilinear grid bounded constraints using the n relational congruences and the $n-1$ extra rectilinear congruences is $\mathrm{O}\big(\nu n^2\big)$, where ν is the number of vertices in $\mathcal{P}$. This is because, for each possible grid bounded constraint, we calculate the value of the constraint at each vertex then take the maximum and minimum of these to be the bounds.

6.6.2 Traditional Integer Programming Methods

An alternative approach to minimising the polyhedron with respect to the grid points is to consider the already well researched topic of integer programming. It is well known that computing the integer hull of a polyhedron is equivalent to solving an integer programming (IP) problem, that is, solve $\max\{\mathbf{c}^{\mathrm{T}}\mathbf{x} | A\mathbf{x} \leq \mathbf{b}\}$ where A is an $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{x} \in \mathbb{Z}^n$. In our case

we do not necessarily have the integer grid $\mathbb{Z}^n$, however we can find the affine transformation that maps the original grid to the integer grid. When this affine transformation is then applied to the original grid and polyhedron we will get a problem that is now an integer programming problem and hence can be solved using the techniques described below. Once our integer programming problem has been solved we can then apply the inverse of the affine transformation to get the grid and polyhedron systems we require.

6.6.2.1 Branch and Bound

The branch and bound method is based on the classical approach of divide and conquer, the algorithm proceeds by splitting the problem into smaller sub-problems and solves their linear programming relaxations to provide upper bounds on the objective value. Although the outline of the algorithm remains the same, the specific details of how the algorithm is to be implemented depends on the problem in hand. The differences include how many sub-problems to create at any given point, which variable the emphasis of the sub-problem will focus, and which sub-problem to tackle first. At this point as we are only concerned with producing a weakly tight polyhedron, our choices for the algorithm should focus more on producing a better approximation quickly rather than a more accurate solution. We will now discuss each of these choices with our problem in mind. For the following descriptions suppose we start with the following IP, $\max\{\mathbf{c}^{\mathrm{T}}\mathbf{x} \in \mathbb{R}^n | A\mathbf{x} \leq \mathbf{b}\}$ where A is an $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{x} \in \mathbb{Z}^n$. Also suppose after applying the simplex method to the LP-relaxation we have a solution $\mathbf{x} = (\chi_1, \ldots, \chi_n)$. Let us first consider the number of sub-problems to create:

- Variable Dichotomy: Suppose the solution to the relaxation has some variable, say χ_i, which is fractional. Then the problem is split into two new sub-problems, one with the extra inequality $x_i \leq \lfloor \chi_i \rfloor$ and the other with the extra inequality $x_i \geq \lceil \chi_i \rceil$.

- Bounded Variables: Suppose the solution to the relaxation has some value, say χ_i, which is fractional and we know that $x_i \in \{s, \ldots, t\}$. Then we can split the problem into $t - s + 1$ sub-problems, each with the extra equality $x_i = j$ for $j \in \{s, \ldots, t\}$.

In our case, the bounds within which a variables values may lay could be large or it is possible that we may not know what the bounds are for each variable, therefore we believe it would be best to use the variable dichotomy method to choose the type of sub-problem. This now leads us to the problem that it may be possible for more than one variable to have a fractional value. Therefore we need a method for choosing the variable the sub-problems will gain the extra inequality in.

- The Most Fractional Variable: Given a variable x_i, we say its fractional value is $\min\{f_i, 1 - f_i\}$, where $f_i = \chi_i - \lfloor \chi_i \rfloor$. Then if V is the set of variables which are fractional, the variable we choose is the one whose fractional value is largest, ie $\max_{i \in V} \min\{f_i, 1 - f_i\}$.

- In order: Choose the first variable which is fractional.

Finally let us consider the problem of deciding which sub-problem to tackle first. It can be seen that the algorithm produces a tree of problems where the initial problem is the root of the tree.

- Depth First with Backtracking: Descend as far down a branch of the tree as possible until we can get no further then move onto the last sub-problem created and continue with that branch.
- Breadth First: Starting at the left consider every sub-problem at the same level of the tree before creating any more sub-problems.
- Best Bound: Choose the branch whose LP-relaxation has the best objective value, therefore in the case of maximisation problems choose the largest.
- Most Fractional: Compute the fractional value, $\min\{f_i, 1 - f_i\}$, for each of the branches, then choose the branch with the maximum fractional value.

At this time as we are only concerned with gaining a better approximation rather than a precise one, it would be best if any implementation we have limits the length of any branches we may create. Therefore if the limit of the length of branches is small we would be best choosing a breadth first search. As within such a small search any advantages in efficiency gained by choosing a best bound or fractional approach will be lost in the extra computations these processes require. If however the branch restriction length is larger (say ≥ 3) we recommend applying the most fractional approach.

Example 6.41 *Consider the example given in [81]. Suppose the (IP) is given by*

$$\begin{aligned} max \quad & 4x_1 - x_2 = z \\ subject\ to \quad & 7x_1 - 2x_2 \leq 14 \\ & x_2 \leq 3 \\ & 2x_1 - 2x_2 \leq 3 \\ & \mathbf{x} \in \mathbb{Z}^2. \end{aligned}$$

Then the LP-relaxation has the solution $\mathbf{x} = (20/7, 3)$ *and upper bound* $z^+ = 59/7$. *We now divide the problem into two sub-problems using the variable dichotomy method. As* $x_1 \notin \mathbb{Z}$ *the first sub-problem* S_1 *will have the extra inequality* $x_1 \leq 2$ *and the second sub-problem* S_2 *will gain the inequality* $x_1 \geq 3$. *For this example we will restrict the length of branches to two, therefore we will apply a breadth first search starting from the left. Consider the* S_1 *branch, solving the new (LP) we get the solution* $\mathbf{x} = (2, 1/2)$ *and the upper bound* $z^+ = 15/2$. *Therefore as* $\mathbf{x} \notin \mathbb{Z}^2$ *we can split the problem again to get the sub-problems* S_{11} *which has the extra inequality* $x_2 \leq 0$ *and* S_{12} *which has the extra inequality* $x_2 \geq 1$. *As we are using a breadth first search we now consider the* S_2 *problem. Solving the new (LP) we find that it is infeasible, so the branch* S_2 *is pruned. Now consider the* S_{11} *problem. The solution to the new (LP) is* $\mathbf{x} = (3/2, 0)$

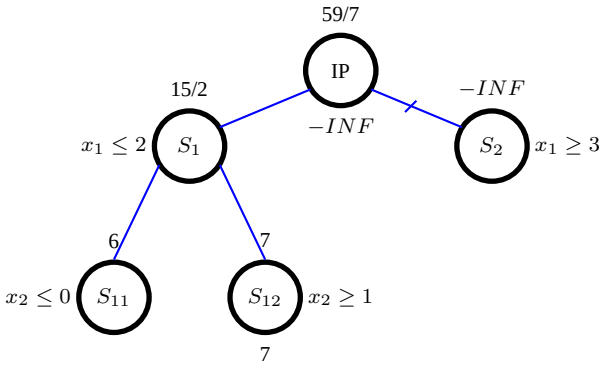


Figure 6.22: The complete branch and bound tree for Example 6.41.

with upper bound $z^+ = 6$, however as we are restricting the length of the branches we will not split the problem again. Therefore consider the final problem S_{12}, we get the solution $\mathbf{x} = (2, 1)$ to the new (LP) which is integral and therefore the upper bound is $z^+ = 7$. The complete branch and bound tree can be seen in Figure 6.22. Now let $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ be the grid-polyhedron where $\mathcal{L} = \mathbb{Z}^2$ and $\mathcal{P} = \text{con}\big(\{4x_1 - x_2 \leq 7, 7x_1 - 2x_2 \leq 14, x_2 \leq 3, 2x_1 - 2x_2 \leq 3\}\big)$. Then the constraint $4x_1 - x_2 \leq 7$ is weakly tight for $\mathcal{H}$.

6.6.2.2 Cutting Planes

The cutting plane method is based on the classical brute force approach, the algorithm successively adds inequalities to the LP problem, called cutting planes, and solves them to hopefully give a better approximation. The classic approach to producing the cutting planes is the Chvátal-Gomory (C-G) procedure and the inequalities produced are called C-G inequalities. This procedure involves taking positive combinations or scalar multiples of the inequalities of the LP and performing integer rounding to generate the new inequalities to be added to the LP so that it can be solved. For example, suppose the inequalities of the LP are given by $A\mathbf{x} \leq \mathbf{b}$ where A is an $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{Z}^n$. Then if $\mathbf{u} \in \mathbb{R}^m_+$ we can produce new inequalities as follows:

1. Multiply the LP inequalities by $\mathbf{u}$. Then

$$\sum_{i=1}^{n} \mathbf{u}^{\mathrm{T}} \mathbf{a}_i x_i \leq \mathbf{u}^{\mathrm{T}} \mathbf{b}$$

   is still a valid inequality.

2. Assuming we have a maximisation problem, we now round down the LHS of the inequalities. Then

$$\sum_{i=1}^{n} \lfloor \mathbf{u}^{\mathrm{T}} \mathbf{a}_i \rfloor x_i \leq \mathbf{u}^{\mathrm{T}} \mathbf{b}$$

is still a valid inequality.

3. We now round down the RHS of the inequalities. Then

$$\sum_{i=1}^{n} \lfloor \mathbf{u}^{\mathrm{T}} \mathbf{a}_i \rfloor x_i \leq \lfloor \mathbf{u}^{\mathrm{T}} \mathbf{b} \rfloor$$

is still a valid inequality and is integer.

Note that it is shown in [67] that this procedure is sufficient to produce all valid inequalities after a finite number of iterations.

Another approach to producing cutting planes is Gomory's fractional cutting plane algorithm. This approach first solves the LP-relaxation to produce the tight solution and from the rows of the associated tight simplex tableau the fractional cuts are taken. For example, suppose the following LP, $\max\{\mathbf{c}^{\mathrm{T}}\mathbf{x} \in \mathbb{R}^n | A\mathbf{x} \leq \mathbf{b}\}$ where A is an $m \times n$ matrix, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{x} \in \mathbb{R}^n$. Also suppose after applying the simplex method to the LP-relaxation we have the tight tableau represented by

$$y_i + \sum_{i=1}^{n} \overline{a}_{ij} v_j = \overline{b}_i \quad \text{for } i = 1, \ldots, m$$

where $\mathbf{y} \in \mathbb{Z}_+^m$ are the basic variables and $\mathbf{v} \in \mathbb{Z}_+^n$ are the non-basic variables. Then the Gomory fractional cutting planes are given by

$$y_i + \sum_{i=1}^{n} \overline{f}_{ij} v_j = \overline{g}_i \quad \text{for } i = 1, \ldots, m$$

where $f_{ij} = \overline{a}_{ij} - \lfloor \overline{a}_{ij} \rfloor$ and $g_i = \overline{b}_i - \lfloor \overline{b}_i \rfloor$.

With both the C-G procedure and Gomory's fractional cuts we believe it would be best for our problem if a set of n cutting planes were added for the first iteration. After that we recommend either stopping or applying the branch and bound technique, since if we were to add another family of n inequalities the problem could become large if n were large.

Example 6.42 *Consider the example given in [55] which uses Gomory's fractional cuts. Suppose we have the (IP)*

$$\begin{aligned} max \quad & 2x_1 + x_2 = z \\ subject\ to \quad & x_1 + 2x_2 \leq 7 \\ & 2x_1 - x_2 \leq 3 \\ & \mathbf{x} \in \mathbb{Z}^2. \end{aligned}$$

Then after applying the simplex algorithm to the LP-relaxation we get the following tight simplex

table.

$$\begin{array}{cccc} & x_4 & x_3 & \\ x_2 & -1/5 & 2/5 & 11/5 \\ x_1 & 2/5 & 1/5 & 13/5 \\ -z & 3/5 & 4/5 & 37/5 \end{array}$$

We can take the fractional parts of each of the rows to create new inequalities. Taking the fractional part of the first row we get $0.4x_3 + 0.8x_4 \geq 0.2$ *which is equivalent to* $x_1 \leq 2.5$*. Similarly, taking the fractional part of the second row we generate the constraint* $0.2x_3 + 0.6x_4 \geq 0.6$ *which is equivalent to* $7x_1 - x_2 \leq 13$*.*

6.7 Related Work

Recall form Section 3.8 that in her thesis Ancourt [1] (see also [68, 71, 72]) considered the domain of $\mathbb{Z}$*-polyhedra*; that is a domain of *integral lattices* intersected with the domain of convex integral polyhedra. Here the product is a direct product and therefore there is no interaction between component domains. Also recall form Section 4.9 that the operations which are similar to our operations are those of grid-polyhedron intersection, affine image and affine pre-image. The operations of grid-polyhedron join and grid-polyhedron difference (as defined here) are not considered; instead the union operator takes two grid-polyhedra $\mathcal{H}_1$ and $\mathcal{H}_2$ and returns a set. The $\mathbb{Z}$-polyhedron domain was then extended in [42] so that the $\mathbb{Z}$-polyhedra are considered to be the affine images of integer polyhedra, where the affine image is the transformation represented by the generator system of the integer lattice.

Example 6.43 *Let* $\mathcal{L} = \mathrm{ggen}(\mathcal{G}_\mathcal{L})$ *and* $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$*, where*

$$\mathcal{G}_\mathcal{L} := \left(\varnothing, \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\right),$$

$$\mathcal{C}_\mathcal{P} := \{x \leq 1,\ y \leq 3\}.$$

Then the $\mathbb{Z}$*-polyhedron is given by the set*

$$\{\mathbf{x} \in \mathcal{L} | x \leq 3,\ y \leq 6\}.$$

Therefore if we think of this interpretation for our grid-polyhedron domain, with the restriction that the generator description for the grids do not contain lines, we consider objects that are affine images of integer polyhedra, where the affine image is the transformation represented by the generator system of the rational grid. The main difference with this interpretation of the $\mathbb{Z}$-polyhedra is how the operations such as intersection, union and difference are now performed, whereas before the operations where applied to the separate components now the desired operation must be applied in stages. First the operation is applied to the lattices to get the new affine transformation,

then this transformation is applied to the $\mathbb{Z}$-polyhedra and finally the operation is applied to the these two new $\mathbb{Z}$-polyhedra.

6.7.1 Products

This section gives an overview of the different ways we could have represented the combining of two domains and considers the advantages and disadvantages of each method.

6.7.1.1 Cartesian Product

The Cartesian product is the most basic of combinations as the product is represented by the pair and there is no interaction between the two domains. Suppose we are given the two abstract domains A_1, A_2 with concretisation functions $\gamma_1 : A_1 \to C$ and $\gamma_2 : A_2 \to C$, respectively. Then the Cartesian product has the domain $A_\times = A_1 \times A_2$. The concretisation function is given by $\gamma : A_\times \to C \times C$, where

$$\gamma\big((a_1, a_2)\big) := \big(\gamma_1(a_1), \gamma_2(a_2)\big)$$

and the abstraction function is given by $\alpha : C \times C \to A_\times$, where

$$\alpha(c, c) := \big(\alpha_1(c), \alpha_2(c)\big).$$

6.7.1.2 Direct Product

The Direct product [28] is the most basic of combinations where the objects are considered to be the intersection of the two components. Like the Cartesian product, for the direct product, there is also no interaction between the two domains. Suppose we are given the two abstract domains A_1, A_2 with concretisation functions $\gamma_1 : A_1 \to C$ and $\gamma_2 : A_2 \to C$, respectively. Then the direct product has the domain $A_\times = A_1 \times A_2$. The concretisation function is given by $\gamma : A_\times \to C$, where

$$\gamma\big((a_1, a_2)\big) := \gamma_1(a_1) \sqcap \gamma_2(a_2)$$

and the abstraction function is given by $\alpha : C \to A_\times$, where

$$\alpha(c) := \big(\alpha_1(c), \alpha_2(c)\big).$$

If the concrete operation $CO : C \to C$ has the corresponding abstract operations AO_1, AO_2 over the abstract domains A_1, A_2, respectively, then the abstract operation over the direct product domain can be constructed as follows

$$AO_\times\big((a_1, a_2)\big) := \big(AO_1(a_1), AO_2(a_2)\big).$$

The advantages of the direct product are that it is easy to implement as the direct product will just pair together the existing implementations. Therefore this product produces the simplest way to gain the extra information not yielded by a single domain analysis. However the disadvantage of the direct product is that since there is no interaction between the elements of each domain an amount of precision can be lost. Also there may be a loss of efficiency, for example, there is no sharing of equalities between the two domain components which could lead to extra operations being performed unnecessarily. Another disadvantage of the direct product is that a Galois insertion is not always formed, which again can lead to a loss of precision.

6.7.1.3 Reduced Product

The Reduced product was introduced by Cousot and Cousot [28] as a way to gain some of the precision lost by the direct product. Suppose we are given the two abstract domains A_1, A_2 with concretisation functions $\gamma_1 : A_1 \to C$ and $\gamma_2 : A_2 \to C$, respectively and the direct product domain $A_\times = A_1 \times A_2$. Then the concept of the reduced product is to add to the direct product a function which maps all the elements with the same concretisation into an equivalence class. Then each class will have an element which represents the class which will be used to improve precision. The reduction function $R : A_\times \to A_\times$ is defined as

$$R\big((a_1, a_2)\big) := \sqcap\big\{(e_1, e_2) \mid \gamma\big((e_1, e_2)\big) = \gamma\big((a_1, a_2)\big)\big\}.$$

Then the reduced product domain is the domain

$$A_R = \big\{R\big((a_1, a_2)\big) \mid a_1 \in A_1, a_2 \in A_2\big\}.$$

The concretisation operator, $\gamma : A_R \to C$, and the abstraction operator, $\alpha : C \to A_R$, are given as follows

$$\gamma\big((a_1, a_2)\big) := \gamma_1(e_1) \sqcap \gamma_2(e_2), \quad \text{where} \quad R\big((a_1, a_2)\big) = (e_1, e_2),$$

$$\alpha(c) := R\Big(\big(\alpha_1(c), \alpha_2(c)\big)\Big).$$

The corresponding abstract operation for the concrete operation $CO : C \to C$ is as follows

$$AO_R\big((a_1, a_2)\big) := R\big(\alpha_1(r), \alpha_2(r)\big) \quad \text{where} \quad r = CO\Big(\gamma\big((a_1, a_2)\big)\Big).$$

The advantages of the reduced product is that it can yield more precise analysis results compared to the direct product and that the reduced product forms a Galois connection provided the two original domains are Galois connections. The disadvantage of the reduced product is that its implementation would require all the abstract operations to be revised with respect to the reduction function. Which is not only a difficult process as it involves the theoretical concretisation function it goes against the fact that we are trying to use existing domains and their abstract operators so

that as little as possible new work needs to be done.

6.7.1.4 Pseudo-reduced Product

The product domain described in [22], called the Pseudo-reduced product by the authors of [25], considers a refined version of the reduced product. The pseudo-reduced product has the domain $A_P = A_R$ and concretisation and abstraction functions follow from those of the reduced product domain. However the abstract operations are defined as

$$AO_P\big((a_1, a_2)\big) := R\big(AO_1(a_1), AO_2(a_2)\big),$$

where AO_1 and AO_2 are the abstract operations over the abstract domains A_1, A_2, respectively, for the concrete operation $CO : C \to C$. As the abstract operations are defined in terms of the reduction function the disadvantages of the Pseudo-reduced product follow from those of the reduced product except that the concretisation function is no longer needed for the abstract operations.

6.7.1.5 Open Product

The Open product is described in [25]. The open product has the domain $A_O = A_1 \times A_2$. The abstract operations are defined as

$$\begin{aligned} &AO_O\big((a_1, a_2)\big) := \\ &\qquad \Big(AO_1\big(Q^1(a_1, a_2), \ldots, Q^m(a_1, a_2)\big)(a_1), AO_2\big(Q^1(a_1, a_2), \ldots, Q^m(a_1, a_2)\big)(a_1)\Big), \end{aligned}$$

where the Q^i are queries, defined as $Q^i(a_1, a_2) = Q^i_1(a_1) \vee Q^i_2(a_2)$, which are monotone functions that map elements or the domain onto tests. The advantage of this domain is that since the abstract domain is that of the Cartesian product the only extra implementation work would be that of producing the query operators. The disadvantage of this domain is that it is not as efficient as the reduced product domain.

6.7.1.6 Granger's Product

Granger introduced his idea of a product domain in [40]. The concept was to have two new operations $\sigma_1 : A_1 \times A_2 \to A_1$ and $\sigma_2 : A_1 \times A_2 \to A_2$ which would refine each of the components of the product thus allowing the two domains to interact. Specifically, σ_1 and σ_2 are such that

$$\begin{aligned} \sigma_1(a_1, a_2) \leq a_1 \quad \text{and} \quad \gamma\Big(\big(\sigma_1(a_1, a_2), a_2\big)\Big) &= \gamma\big((a_1, a_2)\big), \\ \sigma_2(a_1, a_2) \leq a_2 \quad \text{and} \quad \gamma\Big(\big(a_1, \sigma_2(a_1, a_2)\big)\Big) &= \gamma\big((a_1, a_2)\big), \end{aligned}$$

respectively. The product is then defined as the fixpoint of the decreasing iteration sequence given by $\big((\eta_1^n, \eta_2^n)\big)_{n\in\mathbb{N}}$ which is defined as follows

$$\begin{aligned}(\eta_1^0, \eta_2^0) &= (a_1, a_2),\\ (\eta_1^{n+1}, \eta_2^{n+1}) &= \big(\sigma_1(\eta_1^n, \eta_2^n), \sigma_2(\eta_1^n, \eta_2^n)\big).\end{aligned}$$

The advantage of this domain is that since the abstract domain is that of the Direct product the only extra implementation work would be that of producing the refinement operators for each domain. The disadvantage of this domain is that it is not as efficient as the reduced product domain.

6.7.2 Traditional Methods to Test for Emptiness

We now give a description of some alternative ways to test if an integral grid-polyhedron, $\mathbb{Z}^n \cap \mathcal{P}$, is empty.

6.7.2.1 Ellipsoid Method

Khachiyan's method [76, Section 13] and the more general ellipsoid method [67, 76, Section 14] work by finding a series of ellipsoids of decreasing volume and testing if their centres are points within the polyhedron.

Definition 6.44 (Ellipsoid.) *An* ellipsoid *with centre* $\mathbf{y}$ *is the set*

$$E = \{\mathbf{x} \in \mathbb{R}^n | (\mathbf{x} - \mathbf{y})^{\mathrm{T}} D^{-1} (\mathbf{x} - \mathbf{y}) \leq 1\},$$

written as $E(D, \mathbf{y})$*, where* D *is an* $n \times n$ *positive definite matrix and* $\mathbf{y} \in \mathbb{R}^n$*.*

The following outline is taken from [76, Section 13]. Let $\phi = 4n^2\mu$ and $R = 2^\phi$, where μ is the number of constraints in the representation of $\mathcal{P}$. Then $\mathcal{P} \subseteq \{\mathbf{x} | \ \|\mathbf{x}\| \leq R\} = E_0$. The method consists of computing the sequence of ellipsoids $E_0, E_1, E_2, \ldots$ each having a smaller volume, such that $\mathcal{P} \subseteq E_i$, for all i. So for each ellipsoid E_i we have a centre $\mathbf{y}_i$ and a positive definite matrix D_i. Now if the centre, $\mathbf{y}_i$, of the ellipsoid E_i does not lie in $\mathcal{P}$ then it must have violated a constraint of the representation, say $\mathbf{a}^{\mathrm{T}} \cdot \mathbf{x} \leq c$. Then E_{i+1} is the smallest ellipsoid containing $E_i \cap \{\mathbf{x} | \mathbf{a}^{\mathrm{T}} \cdot \mathbf{x} \leq \mathbf{a}^{\mathrm{T}} \cdot \mathbf{y}_i\}$.

6.7.2.2 The Linear Inequality Integer Feasibility Problem

This process consists of checking to see if the unit hypercube will fit inside a bounded polyhedron. To check this property though the polytopes must be of a certain form. The following definitions are used to test for emptiness, taken from [67, Page 515].

Definition 6.45 (Sphere.) *A* sphere *with centre* $\mathbf{y}$ *and radius* r *is the set*

$$S = \{\mathbf{x} \in \mathbb{R}^n | (\mathbf{x} - \mathbf{y})^{\mathrm{T}} (\mathbf{x} - \mathbf{y}) \leq r^2\},$$

written as $S(\mathbf{y}, r)$*, where* $\mathbf{y} \in \mathbb{R}^n$*.*

Definition 6.46 (Round.) *A polytope,* $\mathcal{P} \in \mathbb{R}^n$*, is* round *if there exists a constant* c *and rationals* $\mathbf{y} \in \mathbb{R}^n$*,* $r_1, r_2 \in \mathbb{R}_+$*, such that*

1. $S(\mathbf{y}, r_1) \subseteq \mathcal{P} \subseteq S(\mathbf{y}, r_2)$*;*
2. $\frac{r_2}{r_1} \leq c$*.*

Assuming we have a round full-dimensional polyhedron the test to see if $\mathbb{Z}^n \cap \mathcal{P} \neq \varnothing$ has two cases.

1. $r_1 \geq \frac{1}{2} n^{\frac{1}{2}}$.
   In this case the unit hypercube with centre $\mathbf{y}$ is contained in $\mathcal{P}$ and hence $\mathcal{P}$ must contain an integer point.
2. $r_1 < \frac{1}{2} n^{\frac{1}{2}}$.
   In this case $\mathcal{P}$ can only contain at most one integer point hence the problem can be solved by total enumeration.

The problem occurs with this method if it is found that the polytope is not round. If this happens then an affine transformation must be applied to the polytope to make it round and hence, as noted in [67, Page 518], the problem then becomes equivalent to that of testing for emptiness when we do not have the integral grid.

6.8 Conclusion

We have introduced the partially reduced product of two geometric domains which allows for a range of interaction between the two components. For the product we defined several new reduction operations including the constraint product, weakly tight product and tight product. For the grid-polyhedron domain we gave methods for creating a directed non-redundant congruence, a weakly tight constraint system and a test for emptiness, the last two of which have complexity $\mathrm{O}(n^2\mu)$, where μ is the cardinality of the original constraint system.

Chapter 7

Weakly Relational Grid-Polyhedron Domains

7.1 Introduction

In this chapter we will introduce some weakly relational grid-polyhedron domains. These domains are the product of a grid or a weakly relational grid with some of the weakly relational sub-domains of the polyhedra. Namely we will specify the grid-box domain and introduce the grid-bds, bounded difference grid shape, grid-octagon and ogrid-octagon domains which have not been proposed before. For each of the different combinations of domain we will consider how the result of the weakly tight algorithm, Algorithm 3, is effected. Specifically, we will show, with certain restrictions to the grid, we can achieve results such as a tight or reduced product. We will then consider the effect the restriction to these sub-domains of grid-polyhedra will have on the other operations.

7.2 Grid-Boxes

Let us first consider the combination of a grid with a box.

Definition 7.1 (Grid-Box.) *Let $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ be a box in $\mathbb{CP}_n$ and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ a grid in $\mathbb{G}_n$. Then we say that $\mathcal{H} = (\mathcal{L}, \mathcal{P}) := \mathcal{L} \cap \mathcal{P}$ is a* grid-box. *The* grid-box domain *is a subset of $\mathbb{GP}_n$ and is the set of all grid-boxes in $\mathbb{R}^n$ ordered by the set inclusion relation.*

Note that $\varnothing$ and $\mathbb{R}^n$ are grid-boxes and therefore are the bottom and top elements of the subset respectively.

Definition 7.2 (Rectilinear Grid-Box.) *We say a grid-box is* rectilinear *if and only if the grid is rectilinear.*

Recall that as an n-dimensional *box* $\mathcal{B}$ is a sequence $(I_1, \ldots I_n)$ of intervals over the set $\mathbb{R}$, a 1-dimensional grid-box is a grid-interval. Therefore all the results of this section will also hold for the domain of grid-intervals.

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. As the box domain is a non-relational domain when we create the directed non-redundant congruences from the constraints in $\mathcal{P}$ we will want to create non-relational congruences. Therefore if we only have $\mathcal{L}$ represented by a generator system we can do this using Proposition 4.11 from Section 4.6 as this will create the smallest rectilinear grid containing $\mathcal{L}$, otherwise we will use Algorithm 2. Before we introduce a test for emptiness, we will first discuss how the results of Algorithm 3 are improved when considering grid-boxes. Given a grid-polyhedron $\mathcal{H}$ where $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ and $\mathcal{L}$ is rectilinear then Algorithm 3 will move in the box bounds with respect to the grid so that $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a reduced product. If however $\mathcal{L}$ is not rectilinear then Algorithm 3 will produce a weakly tight box constraint system for $\mathcal{H}$ as shown by Proposition 6.23.

Proposition 7.3 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a rectilinear grid-box where $(\mathcal{L}, \mathcal{P})$ is a constraint and weakly tight product. Then $(\mathcal{L}, \mathcal{P})$ is a reduced product.*

Proof. Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a rectilinear grid-box where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ and $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$. As we can represent any equality $\big(\langle \mathbf{v}, \mathbf{x}\rangle = d\big)$ by the two inequalities $\big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d\big)$ and $\big(\langle \mathbf{v}, \mathbf{x}\rangle \geq d\big)$ we can assume that $\mathcal{C}_\mathcal{P}$ only contains inequalities. Assume that $\mathcal{P}$ is bounded. As $\mathcal{L}$ is the set of vectors in $\mathbb{R}^n$ that satisfy all the congruences of $\mathcal{C}_\mathcal{L}$ we can write $\mathcal{L}$ as

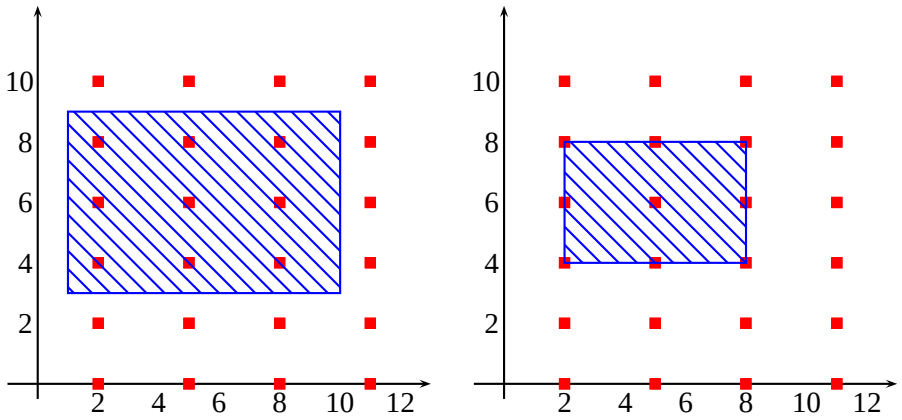
$$\{\mathbf{x} \in \mathbb{R}^n | x_i = t_i + s_i \cdot f_i, \forall s_i \in \mathbb{Z}\}.$$

As $\mathcal{C}_\mathcal{P}$ is a non-relational set of constraints there is $\nu = \big(v_i \cdot x_i \leq d\big) \in \mathcal{C}_\mathcal{P}$, such that $v_i \neq 0$. Then as $(\mathcal{L}, \mathcal{P})$ is a weakly tight product $d = t_i + u_i \cdot f_i$ for some $u_i \in \mathbb{Z}$. Therefore the constraints of $\mathcal{C}_\mathcal{P}$ intersect at grid-box points. If $\mathcal{P}$ is unbounded the result follows. Hence if $(\mathcal{L}, \mathcal{P})$ is a weakly tight product then $(\mathcal{L}, \mathcal{P})$ is a reduced product. □

Corollary 7.4 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a rectilinear grid-box where $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ and $(\mathcal{L}, \mathcal{P})$ is a constraint product. Also let $\mathcal{C}_\mathcal{P}'$ be the constraint system returned by* Algorithm 3 *when applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$. Then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a reduced product.*

Proof. From Proposition 6.23, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a weakly tight product. Then, by Proposition 7.3, $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a reduced product. □

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a rectilinear grid-box where $(\mathcal{L}, \mathcal{P})$ is a constraint product. Then if $\mathcal{C}_\mathcal{P}'$ is the constraint system returned by Algorithm 3 when applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$, $\sigma^1_R(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma^2_R(\mathcal{L}, \mathcal{P}) = \mathrm{con}(\mathcal{C}_\mathcal{P}')$.



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$.

Figure 7.1: Producing a reduced product grid-box.

For any grid-box $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ and $\mathcal{C}_{\mathcal{L}}$ is in minimal form, the cost of performing Algorithm 3, which improves the constraint bounds, depends on a number of factors: if the congruence system $\mathcal{C}_{\mathcal{L}}$ represents a rectilinear grid, then the complexity is $\mathrm{O}\big(n\big)$; if, only the generator system is known and does not represent a rectilinear grid, then the complexity is that of producing the covering box, which is, at worst, $\mathrm{O}\big(n^2\big)$; if, however, the congruence system $\mathcal{C}_{\mathcal{L}}$ does not represent a rectilinear grid, then the complexity is $\mathrm{O}\big(n^3\big)$.

Example 7.5 shows that given a rectilinear grid-box $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ and $(\mathcal{L}, \mathcal{P})$ is not a weakly tight product, Algorithm 3 will return a constraint system $\mathcal{C}_{\mathcal{P}}{}'$ such that the grid-box $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is a reduced product.

Example 7.5 *Consider the grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$, where $\mathcal{C}_{\mathcal{L}} := \{x \equiv_3 2,\ y \equiv_2 0\}$ and the box, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where*

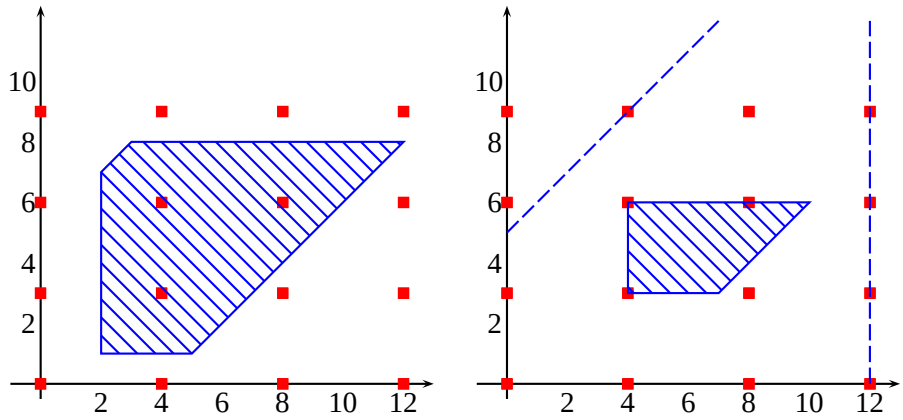
$$\mathcal{C}_{\mathcal{P}} := \{1 \leq x \leq 10,\ 3 \leq y \leq 9\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.1(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight constraint system for $\mathcal{H}$ and also note that $\mathcal{C}_{\mathcal{P}}$ is not a weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$. Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}$, we have $\mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_{\mathcal{P}}{}' := \{2 \leq x \leq 8,\ 4 \leq y \leq 8\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is shown in Figure 7.1(b). Then, not only can it be seen that $\mathcal{C}_{\mathcal{P}}{}'$ is a tight constraint system for $\mathcal{H}$ as every constraint is saturated by at least one grid-box point, but also the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is a reduced product.

For any grid-box $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, where $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ and $\mathcal{C}_{\mathcal{L}}$ is in minimal form, the cost of performing the test for emptiness, depends on a number of factors: if the congruence system $\mathcal{C}_{\mathcal{L}}$ represents a rectilinear grid, then the complexity of testing for emptiness is linear; if, only the



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$.

Figure 7.2: Producing a weakly tight grid-bds.

generator system is known and does not represent a rectilinear grid, then the complexity is that of producing the covering box, which is, at worst, $\mathrm{O}\big(n^2\big)$; if, however, the congruence system $\mathcal{C}_{\mathcal{L}}$ does not represent a rectilinear grid, then the complexity is $\mathrm{O}\big(n^3\big)$.

7.3 Grid-BDS

Let us now consider the grid-bds domain which is a subset of the grid-polyhedron domain and combines the domain of grids with the bounded difference shape domain.

Definition 7.6 (Grid-BDS.) *Let $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ be a bds in $\mathbb{CP}_n$ and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ a grid in $\mathbb{G}_n$. Then we say that $\mathcal{H} = (\mathcal{L}, \mathcal{P}) := \mathcal{L} \cap \mathcal{P}$ is a* grid-bds. *The* grid-bds domain *is a subset of $\mathbb{GP}_n$ and is the set of all grid-bds in $\mathbb{R}^n$ ordered by the set inclusion relation.*

Note that $\varnothing$ and $\mathbb{R}^n$ are grid-bds and therefore are the bottom and top elements of the subset respectively. Recall from Section 2.3.3 that we can represent a bounded difference shape by a weighted graph and that a closed set of constraints for a bds refers to the set derived from a closed weighted graph. As Algorithm 3 will not produce a closed constraint system for the product if we apply the closure algorithm after Algorithm 3 has been performed, we will have a closed constraint system but we will not necessarily have a weakly tight product anymore. Example 7.7 illustrates this point.

Example 7.7 *Consider the grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$, where $\mathcal{C}_{\mathcal{L}} := \{x \equiv_4 0,\ y \equiv_3 0\}$ and the bds, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where*

$$\mathcal{C}_{\mathcal{P}} := \{2 \leq x \leq 12,\ 1 \leq y \leq 8,\ -5 \leq x - y \leq 4\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.2(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight or weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example

$2 \leq x$. *Now after applying* Algorithm 3 *to* $\mathcal{C}_\mathcal{P}$, *we have* $\mathrm{con}(\mathcal{C}_\mathcal{P}')$ *in* $\mathbb{CP}_2$ *where*

$$\mathcal{C}_\mathcal{P}' := \{4 \leq x \leq 12,\ 3 \leq y \leq 6,\ -5 \leq x - y \leq 4\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ *is shown in Figure 7.2(b). Also the constraint* $-5 \leq x - y$ *and* $x \leq 12$ *are illustrated in Figure 7.2(b) by the dashed lines. It can be seen that* Algorithm 3 *does not improve these constraints as they are saturated by the grid point* $(4, 10)^{\mathrm{T}}$ *and* $(12, 0)^{\mathrm{T}}$ *respectively. However, if the closure algorithm were applied to the weighted graph that represents* $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$, *we would get the new closed set of constraints* $\mathrm{closure}(\mathcal{C}_\mathcal{P}') = \mathcal{C}_\mathcal{P}''$ *which are derived from this closed weighted graph, where*

$$\mathcal{C}_\mathcal{P}'' := \{4 \leq x \leq 10,\ 3 \leq y \leq 6,\ -2 \leq x - y \leq 4\}.$$

So by applying the closure algorithm, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}'')\big)$ *and the constraint* $-5 \leq x - y$ *is improved to* $-2 \leq x - y$. *However, now the constraint* $x \leq 12$ *is improved to* $x \leq 10$ *which is not saturated by a grid point. So* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}'')\big)$ *is not a weakly tight product.*

As a bounded difference shape has at most $n^2 + n$ constraints, if $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, then Algorithm 3 has complexity $\mathrm{O}\big(n^2\big)$ if $\mathcal{L}$ is rectilinear, otherwise if the grid is not rectilinear it has complexity $\mathrm{O}\big(n^4\big)$. Also as a bounded difference shape constraint system which is closed is a paired constraint system and as the test for emptiness uses Algorithm 3, it has the same complexity. The following is a corollary to Proposition 7.3.

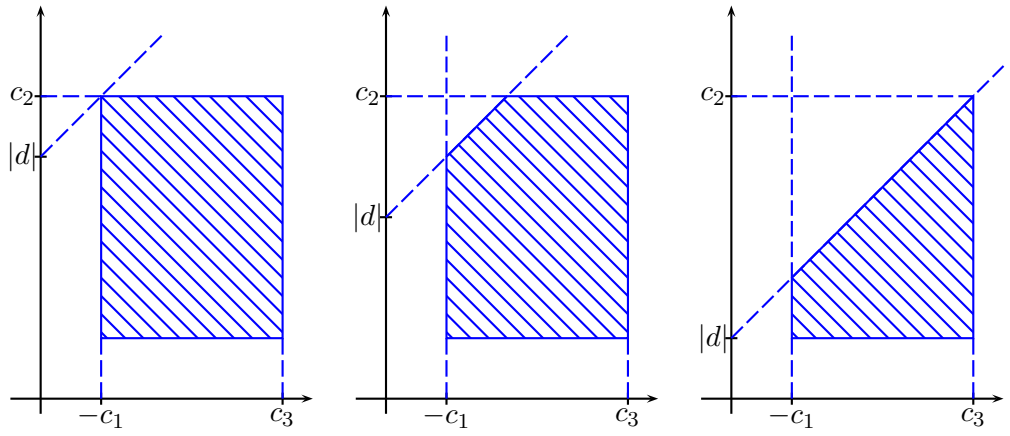
Corollary 7.8 *Let* $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ *be a grid-bds where* $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$, $(\mathcal{L}, \mathcal{P})$ *is a constraint product and* $\mathcal{L}$ *is rectilinear. Also let* $\mathcal{C}_\mathcal{P}'$ *be the bounded difference shape constraint system returned by* Algorithm 3 *when applied to* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$. *If* $\mathcal{C}_\mathcal{P}'$ *is a constraint system that represents a box then* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ *is a reduced product.*

Proof. By Proposition 6.23, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ and $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a weakly tight product. Therefore, by Proposition 7.3, $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a reduced product. □

We will now introduce some results that will be needed to show in certain circumstances a pair representing a grid-bds can be made to be a tight or reduced product.

Proposition 7.9 *Let* $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ *be a grid-bds where* $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ *and* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ *is rectilinear. Also let* $\mathcal{C}_{\mathcal{P}i,j} \subseteq \mathcal{C}_\mathcal{P}$ *be the set of bds constraints over the variables* x_i, x_j, *for* $i, j \in \{1, \ldots, n\}$, *and let* $\mathcal{C}_{\mathcal{L}i,j} \subseteq \mathcal{C}_\mathcal{L}$ *be the congruence system over the variables* x_i, x_j, *for* $i, j \in \{1, \ldots, n\}$. *Then* $(\mathcal{L}, \mathcal{P})$ *is a tight or reduced product if and only if every* 2*-dimensional subset* $\big(\mathrm{gcon}(\mathcal{C}_{\mathcal{L}i,j}), \mathrm{con}(\mathcal{C}_{\mathcal{P}i,j})\big)$ *of* $\big(\mathrm{gcon}(\mathcal{C}_\mathcal{L}), \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$ *is a tight or reduced product.*

Proof. Let us first assume that $(\mathcal{L}, \mathcal{P})$ is a tight product. Then every constraint in $\mathcal{C}_\mathcal{P}$ is saturated by a grid-bds point. Therefore, any constraint in a 2-dimensional subset of $\mathcal{C}_\mathcal{P}$ is saturated by a grid-bds point. Hence, every 2-dimensional subset of $\big(\mathrm{gcon}(\mathcal{C}_\mathcal{L}), \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$ is a tight product.



(a) $d = c_1 + c_2$ (b) $d < c_1 + c_2, |d| \neq c_2 - c_3$ (c) $d < c_1 + c_2, |d| \neq c_2 - c_3$

Figure 7.3: Illustrations for Proposition 7.10.

Now suppose that $(\mathcal{L}, \mathcal{P})$ is a reduced product. Then every vertex of $\mathcal{H}$ is a grid-bds point. So each face of $\mathcal{P}$ is a 2-dimensional bds whose vertices are grid-bds points. Hence, every 2-dimensional subset of $\big(\mathrm{gcon}(\mathcal{C}_\mathcal{L}), \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$ is a reduced product.

Suppose that every 2-dimensional subset of $\big(\mathrm{gcon}(\mathcal{C}_\mathcal{L}), \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$ is a tight product. So every constraint in each subset is saturated by a grid-bds point. Now, the set $\mathcal{C}_\mathcal{P}$ is the union of these subsets of constraints. Also, as $\mathcal{L}$ is rectilinear, the set $\mathcal{C}_\mathcal{L}$ is the union of these subsets of congruences. Thus, $(\mathcal{L}, \mathcal{P})$ is a tight product. Finally, suppose that every 2-dimensional subset of $\big(\mathrm{gcon}(\mathcal{C}_\mathcal{L}), \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$ is a reduced product. So each subset is a 2-dimensional bds whose vertices are grid-bds points. Now, for any $k \neq i, k \neq j$, we have $\mathcal{C}_{\mathcal{P}i,k}$ and $\mathcal{C}_{\mathcal{P}j,k}$ are reduced products. In n-dimensions we can think of $\mathcal{C}_{\mathcal{P}i,j}$ as the bds where the values of each variable x_k are fixed, for $k \neq i, k \neq j$. So, as $\mathcal{L}$ is rectilinear, $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}i,j})\big)$ is a reduced product if $x_k \equiv_{f_k} b_k$ for $k \neq i, k \neq j$. Hence $(\mathcal{L}, \mathcal{P})$ is a reduced product. □

Proposition 7.10 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_2$ be a grid-bds where $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$, $\mathcal{C}_\mathcal{P}$ is a closed constraint system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ is rectilinear. Suppose $\nu = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d\big) \in \mathcal{C}_\mathcal{P}$, where $v_i \neq 0$, $v_j \neq 0$ and $v_i \neq v_j$, $\nu_i = (v_i \cdot x_i \leq c_1) \in \mathcal{C}_\mathcal{P}, \nu_j = (v_j \cdot x_j \leq c_2) \in \mathcal{C}_\mathcal{P}$ and $\overline{\nu}_i = (-v_i \cdot x_i \leq c_3) \in \mathcal{C}_\mathcal{P}$. Then, there are only three cases that can occur:*

1. $d = c_1 + c_2$,
2. $d < c_1 + c_2$ *and* $|d| \neq c_2 - c_3$,
3. $d < c_1 + c_2$ *and* $|d| = c_2 - c_3$.

Proof. By Proposition 7.9 we only need to consider the 2-dimensional case. A version of the 2-dimensional scenario for Case (1) can be seen in Figure 7.3(a), a version of the 2-dimensional scenario for Case (2) can be seen in Figure 7.3(b) and a version of the 2-dimensional scenario

for Case (3) can be seen in Figure 7.3(c). As $\mathcal{C}_\mathcal{P}$ is closed, each constraint must intersect at least one other constraint at a vertex of $\mathcal{P}$. Therefore, ν must intersect ν_j at a point such that $-c_1 \leq x \leq c_3$. Therefore $d = c_1 + c_2$ or $d < c_1 + c_2$ and $|d| \neq c_2 - c_3$ or $|d| = c_2 - c_3$. Let $\overline{\nu}_j = (-v_j \cdot x_j \leq c_4) \in \mathcal{C}_\mathcal{P}$. Then ν must intersect ν_i at a point such that $-c_4 \leq x \leq c_2$. This can be shown using Cases (1), (2) and (3) where the x_j and x_i variables are swapped. □

Definition 7.11 (Common Frequency Grid.) *Let $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ be a rectilinear grid where $\mathcal{C}_\mathcal{L}$ is in minimal form. Suppose that the congruences of $\mathcal{C}_\mathcal{L}$ can be ordered such that for all $\gamma_i = (x_i \equiv_{f_i} b_i) \in \mathcal{C}_\mathcal{L}$, $f_i | f_{i+1}$. Then we say that $\mathcal{L}$ is a common frequency grid.*

For Proposition 7.12 and Proposition 7.15, let $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$, where $\mathcal{C}_\mathcal{P}$ is a closed set of constraints for $\mathcal{P}$, and let $\{\mathcal{C}_{\mathcal{P}1}, \mathcal{C}_{\mathcal{P}2}\}$ be a partition of $\mathcal{C}_\mathcal{P}$ where $\mathcal{C}_{\mathcal{P}1}$ contains the non-relational constraints and $\mathcal{C}_{\mathcal{P}2}$ contains the constraints which are not non-relational. Propositions 7.12 and 7.15 will now show that with certain restrictions on the grid description Algorithm 3 will produce a constraint system which is a tight and reduced product for $\mathcal{H}$ respectively.

Proposition 7.12 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a grid-bds where $(\mathcal{L}, \mathcal{P})$ is a constraint product, $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ where $\mathcal{C}_\mathcal{P}$ is a closed constraint system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ is rectilinear and a common frequency grid. Suppose that the following steps are applied:*

1. Algorithm 3 *returns the constraint system $\mathcal{C}_\mathcal{P}{}'$ when it is applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$,*
2. $\mathrm{closure}(\mathcal{C}_\mathcal{P}{}') = \mathcal{C}_\mathcal{P}{}^c$,
3. Algorithm 3 *returns the constraint system $\mathcal{C}_\mathcal{P}{}''$ when it is applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}^c)\big)$.*

Then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}'')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}'')\big)$ is a tight product.

Proof. Let $\mathcal{C}_\mathcal{P} = \mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2}$. As Algorithm 3 considers each constraint bound, one at a time and independently of the next, we can assume that Algorithm 3 is applied to the constraint system $\mathcal{C}_{\mathcal{P}1}$ first and then applied to $\mathcal{C}_{\mathcal{P}2}$. As $\mathcal{L}$ is a common frequency grid, we can assume that, without loss of generality, $f_i | f_j$ for $i < j$. By Proposition 7.9, as $\mathcal{L}$ is rectilinear, we only need to consider the variables x_i and x_j. Suppose first that $\mathcal{P}$ is bounded. As $\mathcal{C}_\mathcal{P}$ is a closed constraint system, if

$$\nu_i = (v_i \cdot x_i \leq c_1) \in \mathcal{C}_{\mathcal{P}1}, \qquad \nu_j = (v_j \cdot x_j \leq c_2) \in \mathcal{C}_{\mathcal{P}1},$$
$$\overline{\nu}_i = (-v_i \cdot x_i \leq c_3) \in \mathcal{C}_{\mathcal{P}1},$$

where $v_i \neq v_j$, then there is $\nu = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d\big) \in \mathcal{C}_{\mathcal{P}2}$, where $v_i \neq 0$, $v_j \neq 0$.

As $\mathcal{C}_\mathcal{P}$ is a closed constraint system, we need to show $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}'')\big)$ is a tight product by considering the three cases from Proposition 7.10. We will prove this by induction on the number of constraints in $\mathcal{C}_{\mathcal{P}2}$. Let

$$\nu_i' = (v_i \cdot x_i \leq c_1') \in \mathcal{C}_\mathcal{P}{}_1', \qquad \nu_j' = (v_j \cdot x_j \leq c_2') \in \mathcal{C}_\mathcal{P}{}_1',$$
$$\overline{\nu}_i' = (-v_i \cdot x_i \leq c_3') \in \mathcal{C}_\mathcal{P}{}_1', \qquad \nu' = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d'\big) \in \mathcal{C}_\mathcal{P}{}_{2i+1}'$$

be the constraints ν_i, ν_j, $\overline{\nu}_i$ and ν, respectively, after Step (1) is applied, let

$$\nu_i'' = (v_i \cdot x_i \leq c_1'') \in \mathcal{C}_{\mathcal{P}_1}^c, \qquad \nu_j'' = (v_j \cdot x_j \leq c_2'') \in \mathcal{C}_{\mathcal{P}_1}^c,$$
$$\overline{\nu}_i'' = (-v_i \cdot x_i \leq c_3'') \in \mathcal{C}_{\mathcal{P}_1}^c, \qquad \nu'' = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d''\big) \in \mathcal{C}_{\mathcal{P}_{2i+1}}^c$$

be the constraints ν_i', ν_j', $\overline{\nu}_i'$ and ν', respectively, after Step (2) is applied and let

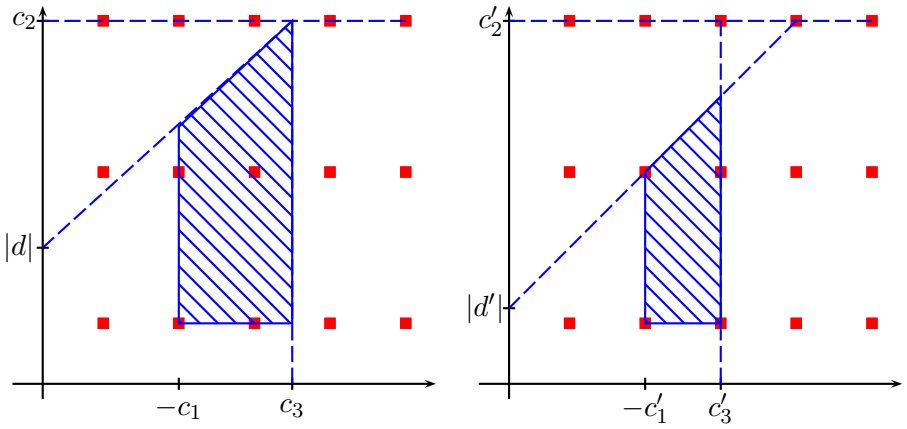
$$\nu_i''' = (v_i \cdot x_i \leq c_1''') \in \mathcal{C}_{\mathcal{P}_1}'', \qquad \nu_j''' = (v_j \cdot x_j \leq c_2''') \in \mathcal{C}_{\mathcal{P}_1}'',$$
$$\overline{\nu}_i''' = (-v_i \cdot x_i \leq c_3''') \in \mathcal{C}_{\mathcal{P}_1}'', \qquad \nu''' = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d'''\big) \in \mathcal{C}_{\mathcal{P}_{2i+1}}''$$

be the constraints ν_i'', ν_j'', $\overline{\nu}_i''$ and ν'', respectively, after Step (3) is applied.

If $\mathcal{C}_{\mathcal{P}2} = \varnothing$, then by Proposition 7.3, as $\mathcal{L}$ is rectilinear, when Step (1) is applied, the grid-box $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}')\big)$ is a reduced product. So the pair of constraints ν_i' and ν_j' intersect at a grid-bds point. Therefore after Step (2) is applied ν_i'' and ν_j'' intersect at a grid-bds point. Therefore if Step (3) returns the constraint system $\mathcal{C}_{\mathcal{P}_1}''$ then the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}'')\big)$ is a reduced product and therefore is a tight product. Let us suppose that the result holds for the set of constraints $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i}$ where $\mathcal{C}_{\mathcal{P}2i} \subseteq \mathcal{C}_{\mathcal{P}2}$. We will now show the result holds for the set of constraints $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1}$ where $\mathcal{C}_{\mathcal{P}2i+1} = \mathcal{C}_{\mathcal{P}2i} \cup \{\nu\}$.

Suppose that we have Case (1) from Proposition 7.10. As $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}^c)\big)$ is a reduced product we have that $d'' = c_1'' + c_2''$. Hence ν'' saturates a grid-bds point. Hence $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}^c \cup \{\nu''\})\big)$ is a tight product. So if Step (3) returns the constraint system $\mathcal{C}_{\mathcal{P}_1}'' \cup \mathcal{C}_{\mathcal{P}_{2i+2}}''$, then $\nu'' = \nu'''$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}2}'' \cup \mathcal{C}_{\mathcal{P}_{2i+1}}'')\big)$ is a tight product.

Suppose now that we have Case (2) from Proposition 7.10. Suppose that Step (1) is applied to $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1}$. If $d' = c_1' + c_2'$, then after Step (2) is applied, $d'' = c_1'' + c_2''$. Hence, from Case (1), ν_i'' and ν_j'' intersect at a grid-bds point and ν'' saturates a grid-bds point. Hence, $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}^c \cup \mathcal{C}_{\mathcal{P}_{2i+1}}^c)\big)$ is a reduced product, and if Step (3) returns the constraint system $\mathcal{C}_{\mathcal{P}_1}'' \cup \mathcal{C}_{\mathcal{P}_{2i+1}}''$, then the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}'' \cup \mathcal{C}_{\mathcal{P}_{2i+1}}'')\big)$ is a tight product. Therefore, suppose that $d' < c_1' + c_2'$ and $d'' < c_1'' + c_2''$. As the grid-box $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}')\big)$ is a reduced product ν_i'' and ν_j'' will be saturated by grid-bds points. Therefore we must show that either ν'' intersects ν_i'' at a grid-bds point or ν'' intersects ν_j'' at a grid-bds point. As $f_i | f_j$, we have that $m = \gcd(f_i, f_j) = f_i$. Also, as $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}_1}' \cup \mathcal{C}_{\mathcal{P}_{2i+1}}')\big)$ is a weakly tight product and $d'' < c_1'' + c_2''$, $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) \neq \varnothing$, where $\nu_e = \big(\langle \mathbf{v}, \mathbf{x} \rangle = d''\big)$. As $(\mathcal{L}, \mathcal{P})$ is a constraint product there are $(v_i \cdot x_i \equiv_{f_i} b_i) \in \mathcal{C}_{\mathcal{L}}$ and $(v_j \cdot x_j \equiv_{f_j} b_j) \in \mathcal{C}_{\mathcal{L}}$. So there is $\beta = \big(\langle \mathbf{v}, \mathbf{x} \rangle \equiv_m t\big) \in \mathcal{C}_{\mathcal{L}}$. Therefore, as $t = b_i + b_j$, we have that $d'' = b_i + b_j + s \cdot f_i$, where $s \in \mathbb{Z}$. Now, as $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) \neq \varnothing$, we have that



(a) $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1})\big)$. (b) $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}'} \cup \mathcal{C}_{\mathcal{P}'2i+1})\big)$.

Figure 7.4: Illustrations for the proof of Proposition 7.12.

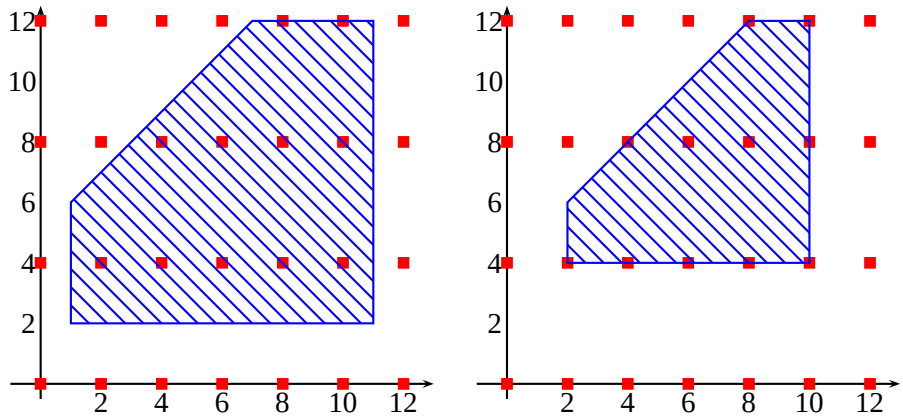
$$\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) =$$

$$\begin{aligned}
&= \big\{\mathbf{x} \in \mathcal{L} \big| \nu_e \cap (v_i \cdot x_i = b_i + s_i \cdot f_i), \forall s_i \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i + v_j \cdot x_j = b_i + b_j + s \cdot f_i) \cap (v_i \cdot x_i = b_i + s_i \cdot f_i), \forall s_i \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (b_i + s_i \cdot f_i + v_j \cdot x_j = b_i + b_j + s \cdot f_i), \forall s_i \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_j \cdot x_j = b_j + (s - s_i) \cdot f_i), \forall s_i \in \mathbb{Z}\big\}.
\end{aligned}$$

Now, as $\mathcal{H} \neq \varnothing$ and $d'' < c_1'' + c_2''$, we know there is a point, $\mathbf{p} \in \mathcal{H}$ such that $v_j \cdot p_j = b_j + s_2 \cdot f_j$ for $s_2 \in \mathbb{Z}$. So all that remains is to show that $\mathbf{p} \in \{\mathbf{x} \in \mathcal{L} | (v_j \cdot x_j = b_j + (s - s_i) \cdot f_i), s_i \in \mathbb{Z}\}$. That is, $(v_j \cdot p_j = b_j + s_2 \cdot f_j = b_j + (s - s_i) \cdot f_i)$. Now this is the same as showing $s_2 \cdot f_j = (s - s_i) \cdot f_i$ and as $f_i | f_j$ there exists $s_j \in \mathbb{Z}$ such that this is true. So ν'' intersects $(v_j \cdot x_j \leq c_1'') \in \mathcal{C}_{\mathcal{P}1}^c$ at a grid-bds point. Hence ν'' saturates a grid-bds point and $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}1}^c \cup \mathcal{C}_{\mathcal{P}2i+1}^c)\big)$ is a tight product. So if Step (3) returns the constraint system $\mathcal{C}_{\mathcal{P}1}'' \cup \mathcal{C}_{\mathcal{P}2i+1}''$, then the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}1}'' \cup \mathcal{C}_{\mathcal{P}2i+1}'')\big)$ is a tight product.

Suppose now we have Case (3) from Proposition 7.10. A version of a 2-dimensional scenario for this case can be seen in Figure 7.4(a) and Figure 7.4(b). If $d' = c_1' + c_2'$ or $d'' = c_1'' + c_2''$ the result follows from the proof for Case (1). If $|d'| = c_2' - c_3'$ then the result follows from the proof of Case (1). Otherwise we have $|d'| \neq c_2' - c_3'$. Then after the Step (2) is applied to $\mathcal{C}_{\mathcal{P}1}' \cup \mathcal{C}_{\mathcal{P}'2i+1}$, $d'' = c_2'' - c_3''$. However, as $f_i | f_j$, ν_j'' does not saturate a grid point anymore. Let Step (3) return the constraint system $\mathcal{C}_{\mathcal{P}1}'' \cup \mathcal{C}_{\mathcal{P}2i+1}''$. Then, by Proposition 7.3 and as $f_i | f_j$, ν_j''' will intersect $\overline{\nu}_i'''$ at a grid-bds point. Also, by Case (2) and as $f_i | f_j$, ν_j''' will intersect ν''' at a grid-bds point. So the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}1}'' \cup \mathcal{C}_{\mathcal{P}'2i+1}'')\big)$ is a tight product. Hence the result follows for all constraints in $\mathcal{C}_{\mathcal{P}2}$.

Now if $\mathcal{P}$ is unbounded then the result follows from Cases (1), (2) and (3). If Steps (1), (2) and (3) are applied to $\mathcal{C}_{\mathcal{P}} = \mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2}$ then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}'')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}'')\big)$ is a



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^{c})\big)$.

Figure 7.5: Producing a tight product grid-bds.

tight product. □

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be the grid-bds where $(\mathcal{L}, \mathcal{P})$ is a constraint product, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ is a closed constraint system, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ is a rectilinear and common frequency grid and $\mathcal{C}_{\mathcal{L}}$ is in minimal form. Let $\mathcal{C}_{\mathcal{P}}{}'$ be the bds constraint system returned by Algorithm 3 when applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}})\big)$, $\mathrm{closure}(\mathcal{C}_{\mathcal{P}}{}') = \mathcal{C}_{\mathcal{P}}{}^{c}$, and $\mathcal{C}_{\mathcal{P}}{}''$ be the bds constraint system returned by Algorithm 3 when applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^{c})\big)$. Then $\sigma^1_T(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma^2_T(\mathcal{L}, \mathcal{P}) = \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')$. Example 7.13 illustrates this when Case (2) from Proposition 7.10 occurs.

Example 7.13 *Consider the grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$, where $\mathcal{C}_{\mathcal{L}} := \{x \equiv_2 0,\ y \equiv_4 0\}$ and the bounded difference shape, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where*

$$\mathcal{C}_{\mathcal{P}} := \{1 \leq x \leq 11,\ 2 \leq y \leq 12,\ -5 \leq x - y \leq 9\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.5(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a weakly tight or tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example $1 \leq x$ is not saturated by a grid point. Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}$, we have $\mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$ in $\mathbb{CP}_2$ where*

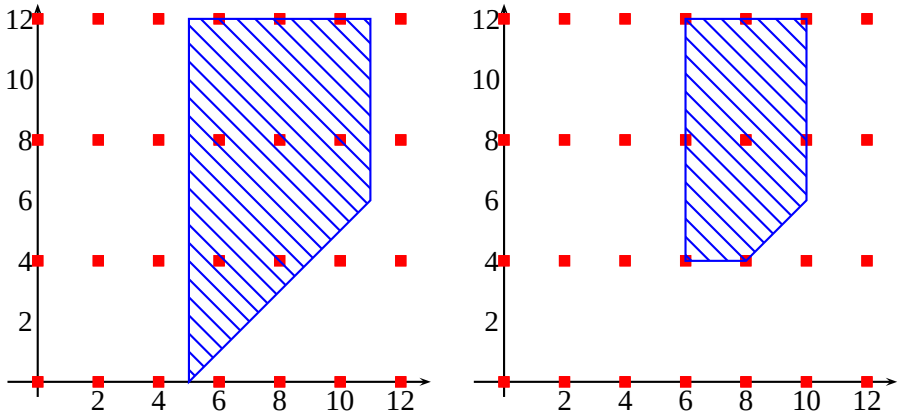
$$\mathcal{C}_{\mathcal{P}}{}' := \{2 \leq x \leq 10,\ 4 \leq y \leq 12,\ -4 \leq x - y \leq 8\}.$$

Let $\mathrm{closure}(\mathcal{C}_{\mathcal{P}}{}') = \mathcal{C}_{\mathcal{P}}{}^{c}$ where

$$\mathcal{C}_{\mathcal{P}}{}^{c} := \{2 \leq x \leq 10,\ 4 \leq y \leq 12,\ -4 \leq x - y \leq 6\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^{c})\big)$ is shown in Figure 7.5(b). Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}{}^{c}$, we have $\mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_{\mathcal{P}}{}'' := \{2 \leq x \leq 10,\ 4 \leq y \leq 12,\ -4 \leq x - y \leq 6\}.$$



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}'')\big)$.

Figure 7.6: Proposition 7.12 requires the condition that $\mathcal{C}_\mathcal{P}$ is a closed constraint system.

Then it can be seen that $\mathcal{C}_\mathcal{P}{}^c = \mathcal{C}_\mathcal{P}{}''$ is a tight constraint system for $\mathcal{H}$ as every constraint is saturated by at least one grid-bds point.

Example 7.14 shows that Proposition 7.12 is successful when Case (3) from Proposition 7.10 occurs.

Example 7.14 *Consider the grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ in $\mathbb{G}_2$, where $\mathcal{C}_\mathcal{L} := \{x \equiv_2 0,\ y \equiv_4 0\}$ and the bounded difference shape, $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ in $\mathbb{CP}_2$, where*

$$\mathcal{C}_\mathcal{P} := \{5 \leq x \leq 11,\ 0 \leq y \leq 12,\ -7 \leq x - y \leq 5\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ is shown in Figure 7.6(a). It can be seen that $\mathcal{C}_\mathcal{P}$ is not a tight or weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example $5 \leq x$ is not saturated by a grid point. Now after applying Algorithm 3 *to $\mathcal{C}_\mathcal{P}$, we have $\mathrm{con}(\mathcal{C}_\mathcal{P}{}')$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_\mathcal{P}{}' := \{6 \leq x \leq 10,\ 0 \leq y \leq 12,\ -6 \leq x - y \leq 4\}.$$

Let $\mathrm{closure}(\mathcal{C}_\mathcal{P}{}') = \mathcal{C}_\mathcal{P}{}^c$ where

$$\mathcal{C}_\mathcal{P}{}^c := \{6 \leq x \leq 10,\ 2 \leq y \leq 12,\ -6 \leq x - y \leq 4\}.$$

Now after applying Algorithm 3 *to $\mathcal{C}_\mathcal{P}{}^c$, we have $\mathrm{con}(\mathcal{C}_\mathcal{P}{}'')$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_\mathcal{P}{}'' := \{6 \leq x \leq 10,\ 4 \leq y \leq 12,\ -6 \leq x - y \leq 4\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}^c)\big)$ is shown in Figure 7.6(b). Then, it can be seen that $\mathcal{C}_\mathcal{P}{}''$ is a a tight constraint system for $\mathcal{H}$.

Proposition 7.15 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a grid-bds where $(\mathcal{L}, \mathcal{P})$ is a constraint product, $\mathcal{P} = \text{con}(\mathcal{C}_\mathcal{P})$ where $\mathcal{C}_\mathcal{P}$ is a closed constraint system and $\mathcal{L} = \text{gcon}(\mathcal{C}_\mathcal{L})$ is a rectilinear grid such that all proper congruences have the same modulus f. Suppose that* Algorithm 3 *returns the constraint system $\mathcal{C}_\mathcal{P}'$ when it is applied to $\big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P})\big)$, Then $\mathcal{H} = \big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P}')\big)$ and the pair $\big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P}')\big)$ is a reduced product.*

Proof. Let $\mathcal{C}_\mathcal{P} = \mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2}$. As Algorithm 3 considers each constraint bound, one at a time and independently of the next, we can assume that Algorithm 3 is applied to the constraint system $\mathcal{C}_{\mathcal{P}1}$ first and then applied to $\mathcal{C}_{\mathcal{P}2}$. By Proposition 7.9, as $\mathcal{L}$ is rectilinear, we only need to consider the variables x_i and x_j. Suppose first that $\mathcal{P}$ is bounded. As $\mathcal{C}_\mathcal{P}$ is a closed constraint system, if

$$\nu_i = (v_i \cdot x_i \leq c_1) \in \mathcal{C}_{\mathcal{P}1} \text{ and}$$
$$\nu_j = (v_j \cdot x_j \leq c_2) \in \mathcal{C}_{\mathcal{P}1}$$

where $v_i \neq v_j$, then there is $\nu = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d\big) \in \mathcal{C}_{\mathcal{P}2}$, where $v_i \neq 0$, $v_j \neq 0$.

As $\mathcal{C}_\mathcal{P}$ is a closed constraint system, we need to show $\big(\mathcal{L}, \text{con}(\mathcal{C}_\mathcal{P}')\big)$ is a reduced product by considering the three cases from Proposition 7.10. We will prove this by induction on the number of constraints in $\mathcal{C}_{\mathcal{P}2}$. If $\mathcal{C}_{\mathcal{P}2} = \varnothing$, then by Proposition 7.3, as $\mathcal{L}$ is rectilinear, when Algorithm 3 is applied, the grid-box $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1}')\big)$ is a reduced product.

Let us suppose that the result holds for the set of constraints $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i}$ where $\mathcal{C}_{\mathcal{P}2i} \subseteq \mathcal{C}_{\mathcal{P}2}$. We will now show the result holds for the set of constraints $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1}$ where $\mathcal{C}_{\mathcal{P}2i+1} = \mathcal{C}_{\mathcal{P}2i} \cup \{\nu\}$. Let

$$\nu_i' = (v_i \cdot x_i \leq c_1') \in \mathcal{C}_{\mathcal{P}1}', \qquad \nu_j' = (v_j \cdot x_j \leq c_2') \in \mathcal{C}_{\mathcal{P}1}',$$
$$\overline{\nu}_i' = (-v_i \cdot x_i \leq c_3') \in \mathcal{C}_{\mathcal{P}1}', \qquad \nu' = \big(\langle \mathbf{v}, \mathbf{x} \rangle \leq d'\big) \in \mathcal{C}_{\mathcal{P}2i+1}'$$

be the constraints ν_i, ν_j, $\overline{\nu}_i$ and ν, respectively, after Algorithm 3 is applied to $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1})\big)$,

Suppose that we have Case (1) from Proposition 7.10. A version of the 2-dimensional scenario for this case can be seen in Figure 7.3(a). As all proper congruences of $\mathcal{C}_\mathcal{L}$ have modulus f and $d' = d - \big((d - t) \mod f\big)$, $d - f < d' \leq d$. Hence, as $\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2}$ is a closed constraint system, $d' \geq c_1' + c_2'$. Therefore as $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1}')\big)$ is a reduced product we have that the pair $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1}' \cup \mathcal{C}_{\mathcal{P}2i+1}')\big)$ is a reduced product.

Now suppose that Case (2) from Proposition 7.10 holds. A version of the 2-dimensional scenario for this case can be seen in Figure 7.3(b). Suppose that Algorithm 3 is applied to $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1})\big)$. If $d' = c_1' + c_2'$, then from Case (1), ν_i', ν_j' and ν' intersect at a grid-bds point. Hence $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1}' \cup \mathcal{C}_{\mathcal{P}2i+1}')\big)$ is a reduced product. Therefore suppose that $d' < c_1' + c_2'$. As the grid-box $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1}')\big)$ is a reduced product ν_i' and ν_j' will be saturated by grid-bds points. Therefore we must show that ν' intersects ν_i' at a grid-bds point and ν' intersects ν_j' at a grid-bds point. As $\big(\mathcal{L}, \text{con}(\mathcal{C}_{\mathcal{P}1}' \cup \mathcal{C}_{\mathcal{P}2i+1}')\big)$ is a weakly tight product and $d' < c_1' + c_2'$, $\mathcal{L} \cap \text{con}\big(\{\nu_e\}\big) \neq \varnothing$,

where $\nu_e = \big(\langle \mathbf{v}, \mathbf{x}\rangle = d''\big)$. As $(\mathcal{L}, \mathcal{P})$ is a constraint product there are $(v_i \cdot x_i \equiv_f b_i) \in \mathcal{C}_\mathcal{L}$ and $(v_j \cdot x_j \equiv_f b_j) \in \mathcal{C}_\mathcal{L}$. So there is $\beta = \big(\langle \mathbf{v}, \mathbf{x}\rangle \equiv_f t\big) \in \mathcal{C}_\mathcal{L}$. Therefore, as $t = b_i + b_j$, we have that $d' = b_i + b_j + s \cdot f$, where $s \in \mathbb{Z}$. Now as $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) \neq \varnothing$ we have that $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) =$

$$
\begin{aligned}
&= \big\{\mathbf{x} \in \mathcal{L} \big| \nu_e \cap (v_i \cdot x_i = b_i + s_i \cdot f), \forall s_i \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i + v_j \cdot x_j = b_i + b_j + s \cdot f) \cap (v_i \cdot x_i = b_i + s_i \cdot f), \forall s_i \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (b_i + s_i \cdot f + v_j \cdot x_j = b_i + b_j + s \cdot f), \forall s_i \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_j \cdot x_j = b_j + (s - s_i) \cdot f), \forall s_i \in \mathbb{Z}\big\}.
\end{aligned}
$$

Now there exists $s_2 \in \mathbb{Z}$ such that $v_j \cdot x_j = b_j + s_2 \cdot f = c'_2$. So, as $d' < c'_1 + c'_2$, ν' intersects $(v_j \cdot x_j \leq c'_2) \in \mathcal{C}_{\mathcal{P}'_1}$ at a grid-bds point. Also $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) =$

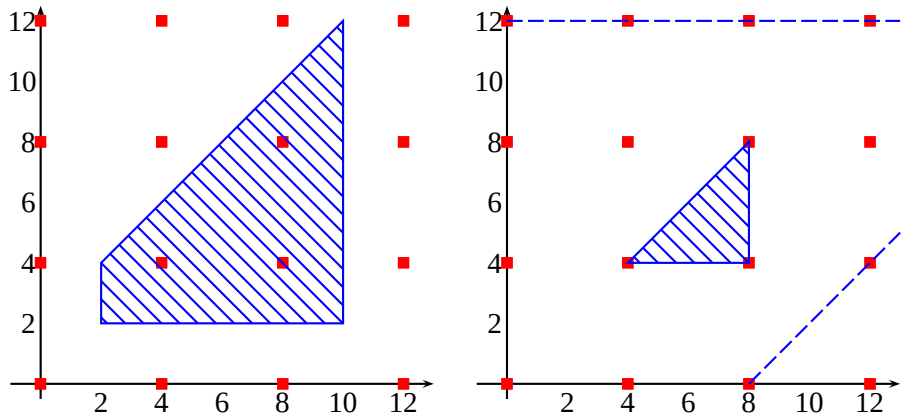
$$
\begin{aligned}
&= \big\{\mathbf{x} \in \mathcal{L} \big| \nu_e \cap (v_j \cdot x_j = b_j + s_j \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_j \cdot x_j + v_i \cdot x_i = b_i + b_j + s \cdot f) \cap (v_j \cdot x_j = b_j + s_j \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (b_j + s_j \cdot f + v_i \cdot x_i = b_i + b_j + s \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i = b_i + (s - s_j) \cdot f), \forall s_j \in \mathbb{Z}\big\}.
\end{aligned}
$$

Now there exists $s_1 \in \mathbb{Z}$ such that $v_i \cdot x_i = b_i + s_1 \cdot f = c'_1$. So, as $d' < c'_1 + c'_2$, ν' intersects $(v_i \cdot x_i \leq c'_1) \in \mathcal{C}_{\mathcal{P}'_1}$ at a grid-bds point. So if Algorithm 3 returns the constraint system $\mathcal{C}_{\mathcal{P}'_1} \cup \mathcal{C}_{\mathcal{P}'_{2i+1}}$ when it is applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}1} \cup \mathcal{C}_{\mathcal{P}2i+1})\big)$, then the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}'_1} \cup \mathcal{C}_{\mathcal{P}'_{2i+1}})\big)$ is a reduced product.

Now suppose that Case (3) from Proposition 7.10 holds. A version of the 2-dimensional scenario for this case can be seen in Figure 7.3(c). If $d' = c'_1 + c'_2$ or $|d'| = c'_2 - c'_3$ the result follows from the proof for Case (1). Otherwise we have $|d'| \neq c'_2 - c'_3$. All that remains to show is that ν' intersects $\overline{\nu}'_i$ at a grid-bds point. As $(\mathcal{L}, \mathcal{P})$ is a constraint product there are $(-v_i \cdot x_i \equiv_f -b_i) \in \mathcal{C}_\mathcal{L}$ and $(v_j \cdot x_j \equiv_f b_j) \in \mathcal{C}_\mathcal{L}$. So there is $\beta = \big(\langle \mathbf{v}, \mathbf{x}\rangle \equiv_f t\big) \in \mathcal{C}_\mathcal{L}$. Therefore as $t = b_i + b_j$, we have that $d' = b_i + b_j + s \cdot f$, where $s \in \mathbb{Z}$. Now as $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) \neq \varnothing$ we have that $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) =$

$$
\begin{aligned}
&= \big\{\mathbf{x} \in \mathcal{L} \big| \nu_e \cap (v_j \cdot x_j = b_j + s_j \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_j \cdot x_j + v_i \cdot x_i = b_i + b_j + s \cdot f) \cap (v_j \cdot x_j = b_j + s_j \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (b_j + s_j \cdot f + v_i \cdot x_i = b_i + b_j + s \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i = b_i + (s - s_j) \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (-v_i \cdot x_i = -b_i - (s - s_j) \cdot f), \forall s_j \in \mathbb{Z}\big\}.
\end{aligned}
$$

Now there exists $s_1 \in \mathbb{Z}$ such that $-v_i \cdot x_i = -b_i - s_1 \cdot f = c'_3$. So, as $d' < c'_1 + c'_2$, ν' intersects $\overline{\nu}'_i$ at a grid-bds point. The fact that ν' will intersect some other constraint at a grid-bds point follows from either this case or Case (2). So the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}'_1} \cup \mathcal{C}_{\mathcal{P}'_{2i+1}})\big)$ is a reduced product. Hence the result follows for all constraints in $\mathcal{C}_{\mathcal{P}2}$.



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$.

Figure 7.7: Producing a reduced product grid-bds.

Now if $\mathcal{P}$ is unbounded the result follows from Cases (1), (2) and (3). Therefore if $\mathcal{C}_{\mathcal{P}}' = \mathcal{C}_{\mathcal{P}1}' \cup \mathcal{C}_{\mathcal{P}2}'$ is the bounded difference shape constraint system returned by Algorithm 3 then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ is a reduced product. □

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be the grid-bds where $(\mathcal{L}, \mathcal{P})$ is a constraint product, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ is a closed constraint system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ is a rectilinear grid such that all proper congruences have the same modulus f. Let $\mathcal{C}_{\mathcal{P}}'$ be the bds constraint system returned by Algorithm 3, then $\sigma_R^1(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma_R^2(\mathcal{L}, \mathcal{P}) = \mathrm{con}(\mathcal{C}_{\mathcal{P}}')$. Example 7.16 demonstrates this.

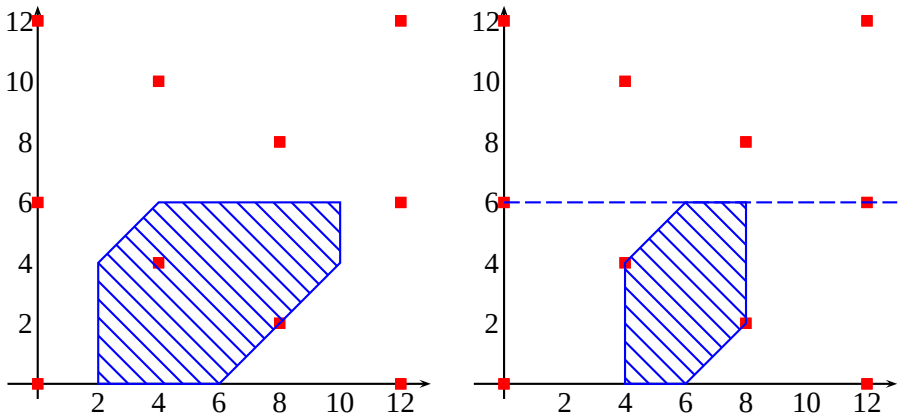
Example 7.16 *Consider the grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$, where $\mathcal{C}_{\mathcal{L}} := \{x \equiv_4 0,\ y \equiv_4 0\}$ and the bounded difference shape, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where*

$$\mathcal{C}_{\mathcal{P}} := \{2 \leq x \leq 10,\ 2 \leq y \leq 12,\ -2 \leq x - y \leq 8\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.7(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight or weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example the constraint $2 \leq x$ is not saturated by a grid point. Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}$, we have $\mathrm{con}(\mathcal{C}_{\mathcal{P}}')$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_{\mathcal{P}}' := \{4 \leq x \leq 8,\ 4 \leq y \leq 12,\ 0 \leq x - y \leq 8\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ is shown in Figure 7.7(b). It can be seen that constraints $y \leq 12$ and $x - y \leq 8$, illustrated by the dashed lines, are not saturated by a grid-bds point, but the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ is a reduced product.



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$.

Figure 7.8: Algorithm 3 does not always produce a reduced product bdgs.

7.3.1 BDGS

Let us now consider the bounded difference grid shape domain which is a subset of the grid-bds domain which takes the product of a bounded difference grid with a bounded difference shape.

Definition 7.17 (BDGS.) *Let $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ be a bounded difference shape in $\mathbb{CP}_n$ and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ a bounded difference grid in $\mathbb{G}_n$. Then we say that $\mathcal{H} = (\mathcal{L}, \mathcal{P}) := \mathcal{L} \cap \mathcal{P}$ is a* bounded difference grid shape (BDGS). *The* bdgs domain *is a subset of $\mathbb{GP}_n$ and is the set of all bounded difference grid shapes in $\mathbb{R}^n$ ordered by the set inclusion relation.*

Note that $\varnothing$ and $\mathbb{R}^n$ are bounded difference grid shapes and therefore are the bottom and top elements of the subset respectively. The results of Corollary 7.8, Proposition 7.12 and Proposition 7.15 hold for a bdgs $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ if $\mathcal{L}$ has a rectilinear representation. Unlike the rectilinear grid-box, if Algorithm 3 is applied to a bdgs $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}})\big)$ and returns $\mathcal{C}_{\mathcal{P}}'$ then the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ is not always a reduced product. Example 7.18 illustrates this.

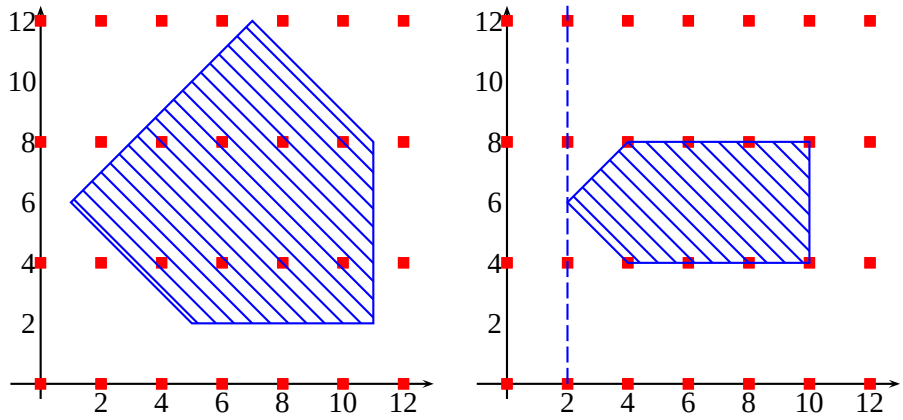
Example 7.18 *Consider the bounded difference grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$, where*

$$\mathcal{C}_{\mathcal{L}} := \{x \equiv_4 0,\ y \equiv_2 0,\ x - y \equiv_6 0\}$$

and the bounded difference shape, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where

$$\mathcal{C}_{\mathcal{P}} := \{2 \leq x \leq 10,\ 0 \leq y \leq 6,\ -2 \leq x - y \leq 6\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.8(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight or weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example the constraint $2 \leq x$ is not saturated by a grid point. Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}$, we have*



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')\big)$.

Figure 7.9: Proposition 7.12 does not hold for grid-octagons.

$\mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$ *in* $\mathbb{CP}_2$ *where*

$$\mathcal{C}_{\mathcal{P}}{}' := \{4 \leq x \leq 8,\ 0 \leq y \leq 6,\ 0 \leq x - y \leq 6\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ *is shown in Figure 7.8(b) and it can be seen that* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ *is not a reduced product. Also the pair* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ *is not even a tight product as the constraint* $y \leq 6$*, illustrated by the dashed line, is not saturated by a bdgs point.*

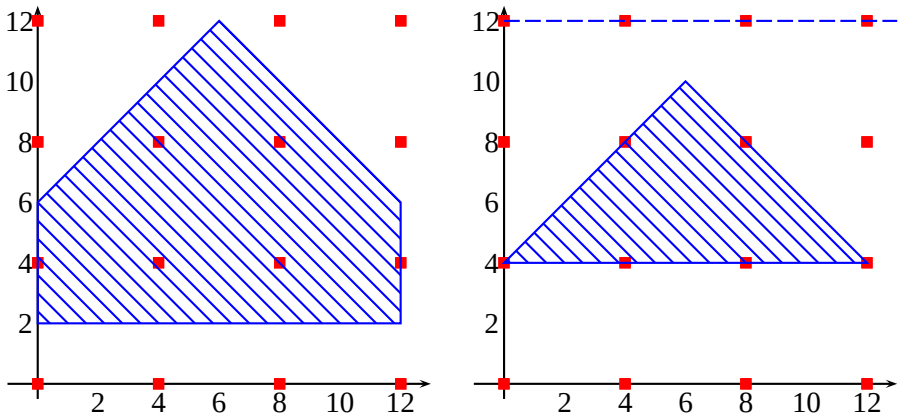
7.4 Grid-Octagons

Let us now consider the grid-octagon domain which is a subset of the grid-polyhedron domain.

Definition 7.19 (Grid-Octagon.) *Let* $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ *be an octagon in* $\mathbb{CP}_n$ *and* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ *a grid in* $\mathbb{G}_n$*. Then we say that* $\mathcal{H} = (\mathcal{L}, \mathcal{P}) := \mathcal{L} \cap \mathcal{P}$ *is a* grid-octagon*. The* grid-octagon domain *is a subset of* $\mathbb{GP}_n$ *and is the set of all grid-octagons in* $\mathbb{R}^n$ *ordered by the set inclusion relation.*

Note that $\varnothing$ and $\mathbb{R}^n$ are grid-octagons, therefore they are the bottom and top elements of the subset respectively. As an octagon has at most $2n^2$ constraints, if $\mathcal{H} = (\mathcal{L}, \mathcal{P})$, then Algorithm 3 has complexity $\mathrm{O}\big(n^2\big)$ if $\mathcal{L}$ is rectilinear, otherwise it has complexity $\mathrm{O}\big(n^4\big)$. Also as a closed octagonal constraint system is a paired constraint system and as the test for emptiness uses Algorithm 3, it has the same complexity.

The result of Corollary 7.8 holds for a grid-octagon $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ if $\mathcal{L}$ has a rectilinear representation, however the results of Proposition 7.12 and Proposition 7.15 do not hold for a grid-octagon $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ if $\mathcal{L}$ has a rectilinear representation. Example 7.20 and Example 7.21, respectively, show this.



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}')\big)$.

Figure 7.10: Proposition 7.15 does not hold for grid-octagons.

Example 7.20 *Consider the grid,* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ *in* $\mathbb{G}_2$*, where* $\mathcal{C}_\mathcal{L} := \{x \equiv_2 0,\ y \equiv_4 0\}$ *and the octagon,* $\mathcal{P} = \mathrm{con}(\mathcal{C}_\mathcal{P})$ *in* $\mathbb{CP}_2$*, where*

$$\mathcal{C}_\mathcal{P} := \{1 \leq x \leq 11,\ 2 \leq y \leq 12,\ -5 \leq x - y \leq 9,\ 7 \leq x + y \leq 19\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ *can be seen in Figure 7.9(a). It can be seen that* $\mathcal{C}_\mathcal{P}$ *is not a tight or weakly tight constraint system for* $\mathcal{H}$ *as not all of the constraints are saturated by a point of* $\mathcal{L}$*, for example the constraint* $1 \leq x$ *is not saturated by a grid point. Now after applying* Algorithm 3 *to* $\mathcal{C}_\mathcal{P}$*, we have* $\mathrm{con}(\mathcal{C}_\mathcal{P}{}')$ *in* $\mathbb{CP}_2$ *where*

$$\mathcal{C}_\mathcal{P}{}' := \{2 \leq x \leq 10,\ 4 \leq y \leq 12,\ -4 \leq x - y \leq 8,\ 8 \leq x + y \leq 18\}.$$

Let $\mathrm{closure}(\mathcal{C}_\mathcal{P}{}') = \mathcal{C}_\mathcal{P}{}^c$ *where*

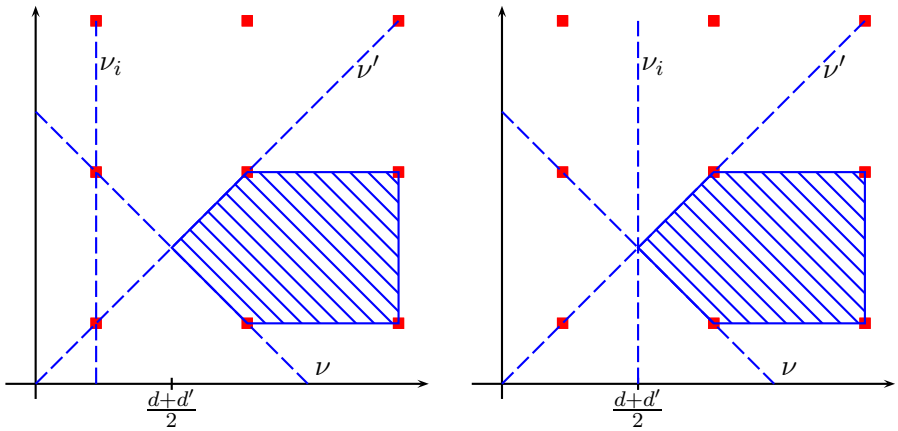
$$\mathcal{C}_\mathcal{P}{}^c := \{2 \leq x \leq 10,\ 4 \leq y \leq 11,\ -4 \leq x - y \leq 6,\ 8 \leq x + y \leq 18\}.$$

Now $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}^c)\big)$ *and the pair* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}^c)\big)$ *is not even a weakly tight product as the constraint* $y \leq 11$ *is not saturated by a grid point. Finally let* $\mathcal{C}_\mathcal{P}{}''$ *be the result of applying* Algorithm 3 *to* $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}^c)\big)$*, where*

$$\mathcal{C}_\mathcal{P}{}'' := \{2 \leq x \leq 10,\ 4 \leq y \leq 8,\ -4 \leq x - y \leq 6,\ 8 \leq x + y \leq 18\}.$$

So $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}'')\big)$ *can be seen in Figure 7.9(b). Now the pair* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}'')\big)$ *is a weakly tight product, but it can be seen in Figure 7.9(b) that the pair is not a tight product as* $2 \leq x$*, illustrated by the dashed line, is not saturated by a grid-octagon point.*

Example 7.21 *Consider the grid,* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_\mathcal{L})$ *in* $\mathbb{G}_2$*, where* $\mathcal{C}_\mathcal{L} := \{x \equiv_4 0,\ y \equiv_4 0\}$ *and the*



(a) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^c)\big)$.

Figure 7.11: Illustrations for the proof of Proposition 7.22.

octagon, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where

$$\mathcal{C}_{\mathcal{P}} := \{0 \leq x \leq 12,\ 2 \leq y \leq 12,\ -6 \leq x - y \leq 10,\ 2 \leq x + y \leq 18\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.10(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight or weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example the constraint $2 \leq y$ is not saturated by a grid point. Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}$, we have $\mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$ in $\mathbb{CP}_2$ where*

$$\mathcal{C}_{\mathcal{P}}{}' := \{0 \leq x \leq 12,\ 4 \leq y \leq 12,\ -4 \leq x - y \leq 8,\ 4 \leq x + y \leq 16\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is shown in Figure 7.10(b) and it can be seen that the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ is not a reduced product or a tight product as the constraint $y \leq 12$, illustrated by the dashed line, is not saturated by a grid-polyhedron point. If we were to apply the closure algorithm to the weighted graph for the octagon represented by $\mathcal{C}_{\mathcal{P}}{}'$ we would get a graph and from this get the constraint system

$$\mathcal{C}_{\mathcal{P}}{}^c := \{0 \leq x \leq 12,\ 4 \leq y \leq 10,\ -4 \leq x - y \leq 8,\ 4 \leq x + y \leq 16\}.$$

Now the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^c)\big)$ is not even a weakly tight product as the constraint $y \leq 10$ is not saturated by a grid point.

Example 7.21 showed that Proposition 7.15 does not hold for grid-octagons even if we apply the closure algorithm. However if we apply Algorithm 3 to $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^c)\big)$ the resulting grid-octagon is a reduced product.

Proposition 7.22 *Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be a non-empty grid-octagon where $(\mathcal{L}, \mathcal{P})$ is a constraint product, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ where $\mathcal{C}_{\mathcal{P}}$ is a closed constraint system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ is a*

grid such that all proper congruences in $\mathcal{C}_\mathcal{L}$ have modulus f. Suppose that the following steps are applied:

1. Algorithm 3 *returns the constraint system $\mathcal{C}_\mathcal{P}'$ when it is applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P})\big)$,*
2. $\mathrm{closure}(\mathcal{C}_\mathcal{P}') = \mathcal{C}_\mathcal{P}{}^c$,
3. Algorithm 3 *returns the constraint system $\mathcal{C}_\mathcal{P}''$ when it is applied to $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}{}^c)\big)$.*

Then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}'')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}'')\big)$ is a reduced product.

Proof. By Proposition 7.9, as $\mathcal{L}$ is rectilinear, we only need to consider the variables x_i and x_j. Let $\mathcal{C}_\mathcal{P}'$ be the constraint system returned after Step (1). Then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ and $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is a weakly tight product. If $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is also a reduced product then the result follows. Therefore suppose that $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ and $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}')\big)$ is not a reduced product.

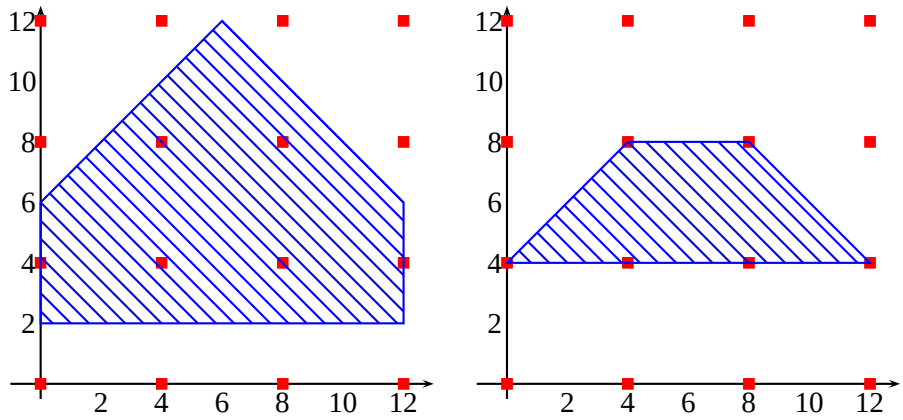
Suppose that $\mathcal{P}$ is bounded. Then there is some vertex of $\mathcal{P}$ that is not a grid-octagon point. As $\mathcal{L}$ is rectilinear, by Proposition 7.15, all the bds constraints intersect at grid-octagon points. Therefore there are constraints of the form

$$\nu = \big(\langle \mathbf{v}, \mathbf{x}\rangle \leq d_1\big) \in \mathcal{C}_\mathcal{P}' \quad \nu' = \big(\langle \mathbf{v}', \mathbf{x}\rangle \leq d_2\big) \in \mathcal{C}_\mathcal{P}'$$

such that ν and ν' do not intersect at a grid-octagon point. Without loss of generality we can assume that $v_i = v_i'$, $v_j \neq v_j'$ and $v_i = v_j$. Figure 7.11(a) illustrates a 2-dimensional version of this case. Then after Step (2) has been performed we will have $\nu, \nu' \in \mathcal{C}_\mathcal{P}{}^c$ and a constraint $\nu_i = (v_i \cdot x_i \leq d_i\big) \in \mathcal{C}_\mathcal{P}{}^c$ such that ν_i does not saturate a grid point and $d_i = \frac{d+d'}{2}$. Figure 7.11(b) illustrates this case. Therefore, after Step (3) is applied we will have $\nu, \nu' \in \mathcal{C}_\mathcal{P}''$, and, as ν_i did not saturate a grid point, $\nu_i' = (v_i \cdot x_i \leq d_i'\big) \in \mathcal{C}_\mathcal{P}''$ such that $d_i' = \frac{d+d'-f}{2}$. As $\mathcal{L}$ is rectilinear, by Proposition 7.15, all the bds constraints of $\mathcal{C}_\mathcal{P}''$ intersect at grid-octagon points. So ν' and ν_i' intersect at a grid-octagon point. Therefore all that remains is to show that ν and ν_i' intersect at a grid-octagon point. As $(\mathcal{L}, \mathcal{P})$ is a constraint product and $\mathcal{P}$ is bounded we have $(v_i \cdot x_i \equiv_f b_i) \in \mathcal{C}_\mathcal{L}$, $(v_j \cdot x_j \equiv_f b_j) \in \mathcal{C}_\mathcal{L}$ and $\big(\langle \mathbf{v}, \mathbf{x}\rangle \equiv_f t\big) \in \mathcal{C}_\mathcal{L}$. As $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}'')\big)$ and $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_\mathcal{P}'')\big)$ is a weakly tight product $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) \neq \varnothing$, where $\nu_e = \big(\langle \mathbf{v}, \mathbf{x}\rangle = d\big)$. Therefore, as $t = b_i + b_j$, we have that $d = b_i + b_j + s \cdot f$, where $s \in \mathbb{Z}$. Now, as $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) \neq \varnothing$, we have that $\mathcal{L} \cap \mathrm{con}\big(\{\nu_e\}\big) =$

$$\begin{aligned}
&= \big\{\mathbf{x} \in \mathcal{L} \big| \nu_e \cap (v_j \cdot x_j = b_j + s_j \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i + v_j \cdot x_j = b_i + b_j + s \cdot f) \cap (v_j \cdot x_j = b_j + s_j \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i + b_j + s_j \cdot f = b_i + b_j + s \cdot f), \forall s_j \in \mathbb{Z}\big\} \\
&= \big\{\mathbf{x} \in \mathcal{L} \big| (v_i \cdot x_i = b_i + (s - s_j) \cdot f), \forall s_j \in \mathbb{Z}\big\}.
\end{aligned}$$

Now there exists $s' \in \mathbb{Z}$ such that $v_i \cdot x_i = b_i + s' \cdot f = d_i'$. So ν intersects $(v_i \cdot x_i \leq d_i') \in \mathcal{C}_\mathcal{P}''$ at a grid-octagon point.



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')\big)$.

Figure 7.12: Producing a reduced product grid-octagon.

If $\mathcal{P}$ is unbounded the result follows from above. Therefore, if Step (1), Step (2) and Step (3) are applied, then $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')\big)$ and the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')\big)$ is a reduced product. □

Let $\mathcal{H} = (\mathcal{L}, \mathcal{P}) \in \mathbb{GP}_n$ be the grid-octagon where $(\mathcal{L}, \mathcal{P})$ is a constraint product, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ is a closed constraint system and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ is a rectilinear grid such that all proper congruences have the same modulus f. Let $\mathcal{C}_{\mathcal{P}}{}'$ be the octagonal constraint system returned by Algorithm 3 when applied to $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}})\big)$, $\mathrm{closure}(\mathcal{C}_{\mathcal{P}}{}') = \mathcal{C}_{\mathcal{P}}{}^{c}$ and $\mathcal{C}_{\mathcal{P}}{}''$ is the constraint system returned by Algorithm 3 when applied to $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^{c})\big)$. Then $\sigma^1_R(\mathcal{L}, \mathcal{P}) = \mathcal{L}$ and $\sigma^2_R(\mathcal{L}, \mathcal{P}) = \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')$. Example 7.23 demonstrates this.

Example 7.23 *Consider the grid-octagon given in Example 7.21 on Page 145, such that* $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ *in* $\mathbb{G}_2$*, where* $\mathcal{C}_{\mathcal{L}} := \{x \equiv_4 0,\ y \equiv_4 0\}$ *and the octagon,* $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ *in* $\mathbb{CP}_2$*, where*

$$\mathcal{C}_{\mathcal{P}} := \{0 \leq x \leq 12,\ 2 \leq y \leq 12,\ -6 \leq x - y \leq 10,\ 2 \leq x + y \leq 18\}.$$

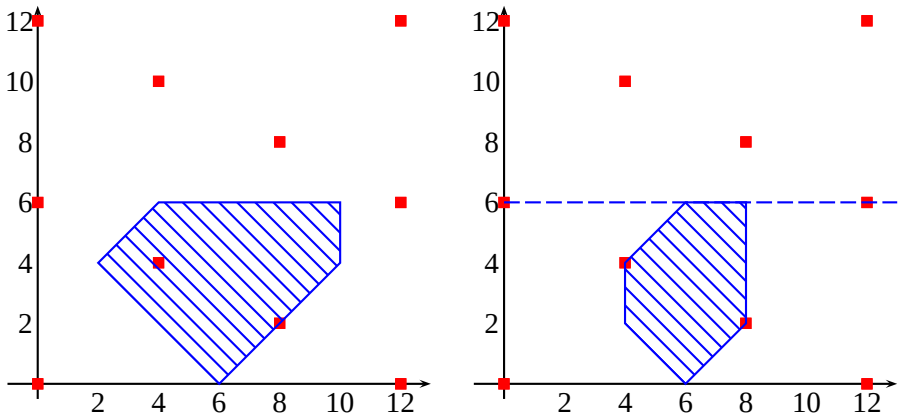
$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ *can be seen in Figure 7.12(a). Recall that* $\mathrm{closure}(\mathcal{C}_{\mathcal{P}}{}') = \mathcal{C}_{\mathcal{P}}{}^{c}$ *where*

$$\mathcal{C}_{\mathcal{P}}{}^{c} := \{0 \leq x \leq 12,\ 4 \leq y \leq 10,\ -4 \leq x - y \leq 8,\ 4 \leq x + y \leq 16\}.$$

Now after applying Algorithm 3 *to* $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}^{c})\big)$*, we get the constraint system*

$$\mathcal{C}_{\mathcal{P}}{}'' := \{0 \leq x \leq 12,\ 4 \leq y \leq 8,\ -4 \leq x - y \leq 8,\ 4 \leq x + y \leq 16\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')\big)$ *is shown in Figure 7.12(b) and it can be seen that the pair* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}'')\big)$ *is a reduced product.*



(a) Grid-polyhedron $\mathcal{H} = (\mathcal{L}, \mathcal{P})$. (b) Grid-polyhedron $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$.

Figure 7.13: Algorithm 3 does not always produce a reduced product ogrid-octagon.

7.4.1 Ogrid-Octagons

Let us now consider the ogrid-octagon domain which is a subset of the grid-octagon domain and whose elements are the product of an octagonal grid and an octagon.

Definition 7.24 (Ogrid-Octagon.) *Let $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ be an octagon in $\mathbb{CP}_n$ and $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ an octagonal grid in $\mathbb{G}_n$. Then we say that $\mathcal{H} = (\mathcal{L}, \mathcal{P}) := \mathcal{L} \cap \mathcal{P}$ is an* ogrid-octagon. *The* ogrid-octagon domain *is a subset of $\mathbb{GP}_n$ and is the set of all octagonal grid-octagons in $\mathbb{R}^n$ ordered by the set inclusion relation.*

Note that $\varnothing$ and $\mathbb{R}^n$ are ogrid-octagons and therefore are the bottom and top elements of the subset respectively. The result of Proposition 7.22 holds for an ogrid-octagon $\mathcal{H} = (\mathcal{L}, \mathcal{P})$ if $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ is rectilinear and all proper congruences of $\mathcal{C}_{\mathcal{L}}$ have modulus f. Unlike the rectilinear grid-box, if Algorithm 3 is applied to an ogrid-octagon $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}})\big)$ and returns $\mathcal{C}_{\mathcal{P}}'$, then, $\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ but the pair $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}')\big)$ is not always a reduced product. Example 7.25 demonstrates this.

Example 7.25 *Consider the octagonal grid, $\mathcal{L} = \mathrm{gcon}(\mathcal{C}_{\mathcal{L}})$ in $\mathbb{G}_2$, where*

$$\mathcal{C}_{\mathcal{L}} := \{x \equiv_4 0,\ y \equiv_2 0,\ x - y \equiv_6 0,\ x + y \equiv_2 0\}$$

and the octagon, $\mathcal{P} = \mathrm{con}(\mathcal{C}_{\mathcal{P}})$ in $\mathbb{CP}_2$, where

$$\mathcal{C}_{\mathcal{P}} := \{2 \leq x \leq 10,\ 0 \leq y \leq 6,\ -2 \leq x - y \leq 6,\ 6 \leq x + y \leq 16\}.$$

$\mathcal{H} = (\mathcal{L}, \mathcal{P})$ can be seen in Figure 7.13(a). It can be seen that $\mathcal{C}_{\mathcal{P}}$ is not a tight or weakly tight constraint system for $\mathcal{H}$ as not all of the constraints are saturated by a point of $\mathcal{L}$, for example the constraint $2 \leq x$ is not saturated by a grid point. Now after applying Algorithm 3 *to $\mathcal{C}_{\mathcal{P}}$, we have*

$\mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')$ *in* $\mathbb{CP}_2$ *where*

$$\mathcal{C}_{\mathcal{P}}{}' := \{4 \leq x \leq 8,\ 0 \leq y \leq 6,\ 0 \leq x - y \leq 6,\ 6 \leq x + y \leq 16\}.$$

$\mathcal{H} = \big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ *is shown in Figure 7.13(b) and it can be seen that the pair* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ *is not a reduced product. The pair* $\big(\mathcal{L}, \mathrm{con}(\mathcal{C}_{\mathcal{P}}{}')\big)$ *is not even a tight product as it can be seen in Figure 7.13(b) that the constraint* $y \leq 6$*, illustrated by the dashed line, is not saturated by an ogrid-octagon point.*

7.5 Operations

If $\mathcal{H}_1$ and $\mathcal{H}_2$ are weakly relational grid-polyhedra then as shown in Sections 7.2 to 7.4, with certain restrictions to the grid, we can produce grid-polyhedra pairs which are reduced products. Therefore since we now have a minimal form for both the grid and polyhedra components, if the grid-polyhedra pairs are reduced products we can easily test if $\mathcal{H}_1 \subseteq \mathcal{H}_2$ or $\mathcal{H}_1 = \mathcal{H}_2$. Also if we have a tight or reduced product we will also know for certain if a weakly relational grid-polyhedron is empty or not.

Finally if $\mathcal{H}_1$ and $\mathcal{H}_2$ are any of the weakly relational grid-polyhedra described in this chapter then, as with the grid-polyhedron case, intersection and difference do not preserve the the reduced product reduction, but the join, affine image and affine pre-image operations do.

7.6 Applications

In this section we discuss applications for the domain of grid-polyhedra and all the sub-domains considered in this chapter.

In [18, 79] an analyser is introduced to detect errors in C programs. They are concerned with checking if arrays are accessed out of bounds and if pointers or variables are accessed without being initialised. The C Global Surveyor (CGS) can either switch between the weakly relational domain of bds and intervals or store the product. It is noted in [18] that future work should include the use of more powerful domains such as the domain of convex polyhedra as this would yield more precise results. Also [33] consider using abstract interpretation to identify buffer overruns in C programs. Here they do use the domain of convex polyhedra to establish the bounds within which the pointer should remain. In [16], Balakrishnan and Reps investigate whether executables such as web-plugins contain or perform harmful operations. Unlike [33], they consider combining pointer analysis and numerical analysis to detect errors in executable programs. They do this by combining an integer interval with an integer rectilinear grid to get a single hybrid object called a *reduced interval congruence (RIC)*. The RIC enables the alignment and stride information to be gathered. The RIC is also considered in [17, 20]. Chouchane et al. consider RICs in the analysis of stack-based operations and in [17], Berstel and Leconte use the RIC in the analysis of programs for Business Rules Management Systems (BRMS). These systems allow businesses

to automate the decisions they make, thus as their marketplace changes the rules system must be updated effectively and error free to enable the business to compete. Therefore the partially reduced grid-polyhedron domain and its sub-domains would also be applicable to all of these problems mentioned.

Following on from [19], Ermedahl et al. [34] also estimate the worst case execution time of a program given a specific system. In order to approximate the WCET the upper bound on the number of loop iterations must be known, this is achieved by slicing the program into subsets using the dependency graph and then the values a variable can take are approximated by a rectilinear integer grid-box. Although not yet studied, it is noted in [34], that a domain such as the grid-polyhedra or one of its weakly relational sub-domains could be used to "allow the size of the abstract states used for loop bound analysis to be minimised", hence more types of loop could be studied.

Separately in [63, 64] and [37] the authors show that integer rectilinear grid-intervals could be used for the analysis of programs. They were concerned with parallelising compilers, specifically data dependence analysis or array reference analysis. That is, to be able to partition a program so that its tasks are performed on separate processors it must be know which elements of an array are referenced and check that two tasks do not try to access the same variable. It was shown in Section 5.5 that the domain of relational grids could be used for this type of analysis. Therefore the domain of grid-polyhedra and its sub-domains could also be used in this way.

7.7 Related Work

In [37, Section 6], Granger considers the reduced product of an integer rectilinear grid-interval and gives a reduction operator which is equivalent to our own. Ermedahl et al. [34] also consider the product of an integer rectilinear grid with an integer interval and we assume that they also use this reduction (although it is not stated) as this work and previous [19] builds on that of the early Granger work [37]. In [51], Miné considers the reduced product domain and states that products of weakly relational base domains could be considered provided they satisfy the acceptable base hypothesis. An example of the reduction operation is given for the grid-interval case which is equivalent to our own for grid-boxes however it is stated in [51] and shown in [53] that the grid-interval domain does not satisfy the acceptable base hypothesis for intersection, that is, the grid-interval domain does not satisfy the condition

$$\bigcap_{i=1}^{n}(\mathcal{L}_i, \mathcal{P}_i) = \varnothing \Rightarrow \exists i, j \in \{1, \ldots, n\}, (\mathcal{L}_i, \mathcal{P}_i) \cap (\mathcal{L}_j, \mathcal{P}_j) = \varnothing.$$

An alternative to considering a product of a rectilinear grid with an interval is to merge the two into one hybrid domain which is considered in several papers. In [64] the authors consider extending the interval domain over $\mathbb{R}$ by assimilating it with a single non-relational congruence, the result is called the *modulo interval*, and written $[a, b]_{m(t)}$ where m is the modulus and t is

	Polyhedron Type			
	Any	Octagon	BDS	Box
Any Grid	Proposition 6.23 $\mathrm{O}(n^2\mu)$	Proposition 6.23 $\mathrm{O}(n^4)$	Proposition 6.23 $\mathrm{O}(n^4)$	Proposition 6.23 $\mathrm{O}(n^3)$
Ogrid	Proposition 6.23 $\mathrm{O}(n^2\mu)$	Proposition 6.23 $\mathrm{O}(n^4)$	Proposition 6.23 $\mathrm{O}(n^4)$	Proposition 6.23 $\mathrm{O}(n^3)$
BDG	Proposition 6.23 $\mathrm{O}(n^2\mu)$	Proposition 6.23 $\mathrm{O}(n^4)$	Proposition 6.23 $\mathrm{O}(n^4)$	Proposition 6.23 $\mathrm{O}(n^3)$
Rectilinear Grid	Proposition 6.23 $\mathrm{O}(n\mu)$	Proposition 6.23 $\mathrm{O}(n^2)$	Proposition 6.23 $\mathrm{O}(n^2)$	Proposition 6.23 $\mathrm{O}(n)$

Table 7.1: Weakly tight polynomial algorithms and complexities.

	Polyhedron Type			
	Any	Octagon	BDS	Box
Rectilinear Grid			Proposition 7.12 $\mathrm{O}(n^2)$	Corollary 7.4 $\mathrm{O}(n)$

Table 7.2: Tight product polynomial algorithms and complexities.

the inhomogeneous term. The modulo interval extends the normal interval domain and includes both set operations and arithmetic operations. However the intersection operation is only considered for two intervals with the same modulus. This was then improved on in [63] where the intersection considers intervals with different modulus. It is noted in both papers that a modulo interval $[a,b]_{m(t)}$ is normalised (a reduced product) if $a \equiv_m t, b \equiv_m t$ and $0 \leq t < m$ although exactly how to calculate the normalised modulo interval is not shown. Balakrishnan and Reps [16] consider the *reduced interval congruence (RIC)* which is a fusion of integer intervals and grids. An RIC is given by a tuple (m,a,b,t) which stands for the set $\{x \equiv_m t | x \in [a,b]\}$. The RIC is assumed to be in minimal form but the description of how to do this is not given. Reps et al. [73] have also defined the *k-bit strided interval*, a triple $m[a,b]$ which represents the set $\{x \in [-2^k, 2^k-1] | a \leq x \leq b, x \equiv_m a\}$. A strided interval is said to be reduced if $b \equiv_m a$ and descriptions of how to compute the addition, subtraction, bitwise and, bitwise or and bitwise negation of strided intervals are given.

7.8 Conclusion

In this chapter we introduced the grid-box domain, grid-bds domain, bounded difference grid shape domain, grid-octagon domain and the ogrid-octagon domain. For each of these five domains we showed that using procedures involving the weakly tight algorithm, Algorithm 3, and the closure algorithm under what circumstances we could produce a tight or reduced product rather than weakly tight product. Specifically for the grid-box domain we showed that if the grid

	Polyhedron Type			
	Any	Octagon	BDS	Box
Rectilinear Grid		Proposition 7.22 $\mathrm{O}(n^2)$	Proposition 7.15 $\mathrm{O}(n^2)$	Corollary 7.4 $\mathrm{O}(n)$

Table 7.3: Reduced product polynomial algorithms and complexities.

is rectilinear we can produce a reduced product grid-box with complexity $\mathrm{O}(n)$. For a grid-bds or bdgs we showed in some circumstances we can produce tight and reduced products with complexity $\mathrm{O}(n^2)$ and for a grid-octagon or ogrid-octagon we showed in some circumstances we can produce a reduced product with complexity $\mathrm{O}(n^2)$. Table 7.1 shows for which combinations of grid and polyhedron we can produce a weakly tight product, Table 7.2 shows for which combinations of grid and polyhedron we can produce a tight product and Table 7.3 shows for which combinations of grid and polyhedron we can produce a reduced product. All three tables also show where the procedure to compute this product is given and what the complexity would be to compute it. Also from these reduced product cases we showed that we now have an exact test for emptiness and comparison.

Chapter 8

Conclusion and Future Work

In this thesis we have presented the domain of Grids. A domain which interprets distribution information about a program or system. We have shown that a grid may be represented by either a set of congruences or a set of generators. For the grid domain we have specified and provided algorithms that minimise the representation of a grid (showing either we can minimise the cardinality of the set or we can create a strong minimal form), convert between representations, create a homogeneous form, perform comparison, test for equality, perform intersection, affine image and pre-image, and widening for both representation. We have also specified and provided algorithms for performing join, difference and covering box which have not been given in previous works [38, 39, 71, 72]. Also for all of these operations we have shown that we achieve complexities better than or equal to previous proposals [38, 39, 61, 62, 71, 72]. In Chapter 5 we have defined two weakly relational grid domains. The bounded difference grid domain is based on the zone-congruence domain by Miné [51, 53] and the octagonal grid is an extension of the zone-congruence domain which encodes information in the way that the octagon domain does [54].

The second topic of the thesis investigates the Grid-Polyhedron domain and many of its subdomains. We introduced the partially reduced product of two geometric domains which allows for a range of interaction between the two components. For this product we specified six reduction operators, namely the direct, reduced, smash, constraint, weakly tight and tight products. For the grid-polyhedron domain we provided operations and algorithms that produce a weakly tight constraint system and test for emptiness, as well providing a complete set of abstract operations. We then introduced the domains grid-box, grid-bds, bdgs, grid-octagon and ogrid-octagon. For each of these domains we showed under what circumstances the weakly tight algorithm will produce tight or reduced products rather than weakly tight products and that this algorithm has a polynomial complexity, see Tables 7.1, 7.2 and 7.3. Specifically, if the grid was rectilinear,

we showed for the grid-box domain that we can produce a reduced product. For a grid-bds or bdgs we showed with restrictions to the grid congruence representation we can produce tight and reduced products and for a grid-octagon or ogrid-octagon we showed with restrictions to the grid congruence representation we can produce a reduced product. Also from these reduced product cases we showed we now have an exact test for emptiness and comparison.

8.1 Future Work

If we had more time we would like to further explore the grid-bds and grid-octagon domains as these domains ensure that any operations can have polynomial complexity. Also we would like to consider other weakly relational grid-polyhedron domains such as a grid-tvpi domain as, in theory, this domain would also ensure that any operations would have polynomial complexity. We have shown that under certain circumstances Algorithm 3, the weakly tight reduction, can produce tight and even reduced products and we believe it is likely that there could be other circumstances for which that is true. We would also like to investigate whether or not it is possible to specify a targeted reduction algorithm for each of the domains so that we can achieve a tight or reduced product in all circumstances.

Also if we had more time we would also like to investigate, for each domain, whether Algorithm 3 could be modified in any way to improve the test for emptiness or more importantly find a point within a grid-polyhedron.

Finally we are interested to know whether is is possible to derive an algorithm for reduction which uses either the grid generator description or the polyhedron generator description as our current algorithm assumes that they must be represented by the congruence and constraint representations respectively.

Bibliography

[1] C. Ancourt. *Génération Automatique de Codes de Transfert pour Multiprocesseurs à Mémoires Locales*. PhD thesis, Université de Paris VI, March 1991.

[2] James A. Anderson. *Discrete Mathematics with Combinatorics*. Prentice Hall, 2001.

[3] R. Bagnara, K. Dobson, P. M. Hill, M. Mundell, and E. Zaffanella. Grids: A domain for analyzing the distribution of numerical values. In G. Puebla, editor, *Logic-based Program Synthesis and Transformation, 16th International Symposium*, volume 4407 of *Lecture Notes in Computer Science*, pages 219–235, Venice, Italy, 2007. Springer-Verlag, Berlin.

[4] R. Bagnara, P. M. Hill, E. Mazzi, and E. Zaffanella. Widening operators for weakly-relational numeric abstractions. In C. Hankin and I. Silveroni, editors, *Static Analysis: Proceedings of the 12th International Symposium*, volume 3672 of *Lecture Notes in Computer Science*, pages 3–18, London, UK, 2005. Springer-Verlag, Berlin.

[5] R. Bagnara, P. M. Hill, E. Mazzi, and E. Zaffanella. Widening operators for weakly-relational numeric abstractions. Quaderno 399, Dipartimento di Matematica, Università di Parma, Italy, 2005. Available at `http://www.cs.unipr.it/Publications/`.

[6] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In R. Cousot, editor, *Static Analysis: Proceedings of the 10th International Symposium*, volume 2694 of *Lecture Notes in Computer Science*, pages 337–354, San Diego, California, USA, 2003. Springer-Verlag, Berlin.

[7] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. *Science of Computer Programming*, 58(1–2):28–56, 2005.

[8] R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. In B. Steffen and G. Levi, editors, *Verification, Model Checking and Abstract Interpretation: Proceedings of the 5th International Conference (VMCAI 2004)*, volume 2937 of *Lecture Notes in Computer Science*, pages 135–148, Venice, Italy, 2003. Springer-Verlag, Berlin.

[9] R. Bagnara, P. M. Hill, and E. Zaffanella. Not necessarily closed convex polyhedra and the double description method. *Formal Aspects of Computing*, 17(2):222–257, 2005.

[10] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. Quaderno 457, Dipartimento di Matematica, Università di Parma, Italy, 2006. Available at `http://www.cs.unipr.it/Publications/`. Also published as `arXiv:cs.MS/0612085`, available from `http://arxiv.org/`.

[11] R. Bagnara, P. M. Hill, and E. Zaffanella. *The Parma Polyhedra Library User's Manual*. Department of Mathematics, University of Parma, Parma, Italy, release 0.9 edition, March 2006. Available at `http://www.cs.unipr.it/ppl/`.

[12] R. Bagnara, P. M. Hill, and E. Zaffanella. An improved tight closure algorithm for integer octagonal constraints. Quaderno 467, Dipartimento di Matematica, Università di Parma, Italy, 2007. Available at `http://www.cs.unipr.it/Publications/`. Also published as `arXiv:0705.4618v2 [cs.DS]`, available from `http://arxiv.org/`.

[13] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 2008. To appear. Journal version of [10].

[14] R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 213–229, Madrid, Spain, 2002. Springer-Verlag, Berlin.

[15] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. An improved tight closure algorithm for integer octagonal constraints. In Francesco Logozzo, Doron Peled, and Lenore D. Zuck, editors, *VMCAI*, volume 4905 of *Lecture Notes in Computer Science*, pages 8–21. Springer, 2008.

[16] G. Balakrishnan and T. W. Reps. Analyzing memory accesses in x86 executables. In *CC*, volume 2985 of *Lecture Notes in Computer Science*, pages 5–23. Springer-Verlag Berlin Heidelberg, 2004.

[17] B. Berstel and M. Leconte. Extending a CP solver with congruences as domains for program verification. In *Workshop on Constraints in Software Testing, Verification and Analysis*, 2006.

[18] G. Brat and A. Venet. Precise and scalable static program analysis of NASA flight software. In *Proceedings of IEEE Aerospace Conference*, pages 1–10, Big Sky, MT, USA, 2005.

[19] S. Bygde. Abstract interpretation and abstract domains. Master's thesis, Mälardalen University, 2006.

[20] M. R. Chouchane, Md. E. Karim, A. Lakhotia, and M. Venable. Analyzing memory accesses in obfuscated x86 executables. In *Conference on Detection of Intrusions and Malware and*

Vulnerability Assessment (DIMVA), volume 3548 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag Berlin Heidelberg, 2005.

[21] Robert Clarisó and Jordi Cortadella. The octahedron abstract domain. In Giacobazzi [35], pages 312–327.

[22] M. Codish, A. Mulkers, M. Bruynooghe, M. Garcìa de la Banda, and M. Hermenegildo. Improving abstract interpretations by combining domains. In *Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 194–205, Copenhagen, Denmark, 1993. ACM Press. Also available as Technical Report CW 162, Department of Computer Science, K.U. Leuven, December 1992.

[23] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[24] T. H. Cormen, T. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[25] A. Cortesi, B. Le Charlier, and P. Van Hentenryck. Combinations of abstract domains for logic programming: Open product and generic pattern construction. *Science of Computer Programming*, 38(1–3):27–71, 2000.

[26] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In B. Robinet, editor, *Proceedings of the Second International Symposium on Programming*, pages 106–130, Paris, France, 1976. Dunod, Paris, France.

[27] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM Press.

[28] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Programming Languages*, pages 269–282, New York, 1979. ACM Press.

[29] P. Cousot and R. Cousot. Abstract interpretation and applications to logic programs. *Journal of Logic Programming*, 13(2&3):103–179, 1992.

[30] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.

[31] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proceedings of the 4th International Symposium on Programming Language Implementation and Logic*

Programming, volume 631 of *Lecture Notes in Computer Science*, pages 269–295, Leuven, Belgium, 1992. Springer-Verlag, Berlin.

[32] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 84–96, Tucson, Arizona, 1978. ACM Press.

[33] N. Dor, M. Rodeh, and S. Sagiv. CSSV: Towards a realistic tool for statically detecting all buffer overflows in C. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03)*, pages 155–167. ACM Press, 2003.

[34] A. Ermedahl, C. Sandberg, J. Gustafsson, S. Bygde, and B. Lisper. Loop bound analysis based on a combination of program slicing, abstract interpretation, and invariant analysis. In *Seventh International Workshop on Worst-Case Execution Time Analysis, (WCET'2007)*, Pisa, Italy, July 2007.

[35] R. Giacobazzi, editor. *Static Analysis: Proceedings of the 11th International Symposium*, volume 3148 of *Lecture Notes in Computer Science*, Verona, Italy, 2004. Springer-Verlag, Berlin.

[36] D. Gopan, F. DiMaio, N. Dor, T. Reps, and M. Sagiv. Numeric domains with summarized dimensions. In K. Jensen and A. Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004*, volume 2988 of *Lecture Notes in Computer Science*, pages 512–529, Barcelona, Spain, 2004. Springer-Verlag, Berlin.

[37] P. Granger. Static analysis of arithmetical congruences. *International Journal of Computer Mathematics*, 30:165–190, 1989.

[38] P. Granger. *Analyses Sémantiques de Congruence*. PhD thesis, École Polytechnique, 921128 Palaiseau, France, July 1991.

[39] P. Granger. Static analysis of linear congruence equalities among variables of a program. In S. Abramsky and T. S. E. Maibaum, editors, *TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Volume 1: Colloquium on Trees in Algebra and Programming (CAAP'91)*, volume 493 of *Lecture Notes in Computer Science*, pages 169–192, Brighton, UK, 1991. Springer-Verlag, Berlin.

[40] P. Granger. Improving the results of static analyses programs by local decreasing iteration. In R. K. Shyamasundar, editor, *Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 68–79, New Delhi, India, 1992. Springer-Verlag, Berlin.

[41] P. Granger. Static analyses of congruence properties on rational numbers (extended abstract). In P. Van Hentenryck, editor, *Static Analysis: Proceedings of the 4th International Symposium*, volume 1302 of *Lecture Notes in Computer Science*, pages 278–292, Paris, France, 1997. Springer-Verlag, Berlin.

[42] Gautam Gupta and Sanjay V. Rajopadhye. The z-polyhedral model. In Katherine A. Yelick and John M. Mellor-Crummey, editors, *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPOPP 2007, San Jose, California, USA*, pages 237–248. ACM, 2007.

[43] N. Halbwachs. *Détermination Automatique de Relations Linéaires Vérifiées par les Variables d'un Programme*. Thèse de 3ème cycle d'informatique, Université scientifique et médicale de Grenoble, Grenoble, France, March 1979.

[44] N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.

[45] M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133–151, 1976.

[46] Andy King and Harald Sondergaard. Inferring Congruence Equations using SAT. In Aarti Gupta and Sharad Malik, editors, *Twentieth International Conference on Computer-Aided Verification*, Lecture Notes in Computer Science. Springer-Verlag, July 2008.

[47] S. Larsen, E. Witchel, and S. P. Amarasinghe. Increasing and detecting memory address congruence. In *Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques (PACT'02)*, pages 18–29, Charlottesville, VA, USA, 2002. IEEE Computer Society Press.

[48] V. Loechner. *PolyLib*: A library for manipulating parameterized polyhedra. Available at `http://icps.u-strasbg.fr/~loechner/polylib/`, March 1999. Declares itself to be a continuation of [80].

[49] A. Miné. A new numerical abstract domain based on difference-bound matrices. In O. Danvy and A. Filinski, editors, *Proceedings of the 2nd Symposium on Programs as Data Objects (PADO 2001)*, volume 2053 of *Lecture Notes in Computer Science*, pages 155–172, Aarhus, Denmark, 2001. Springer-Verlag, Berlin.

[50] A. Miné. The octagon abstract domain. In *Proceedings of the Eighth Working Conference on Reverse Engineering (WCRE'01)*, pages 310–319, Stuttgart, Germany, 2001. IEEE Computer Society Press.

[51] A. Miné. A few graph-based relational numerical abstract domains. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*,

volume 2477 of *Lecture Notes in Computer Science*, pages 117–132, Madrid, Spain, 2002. Springer-Verlag, Berlin.

[52] A. Miné. *The Octagon Abstract Domain Library*. Semantics and Abstract Interpretation Computer Science Lab., École Normale Supérieure, Paris, France, release 0.9.6 edition, October 2002. Available at `http://www.di.ens.fr/~mine/oct/`.

[53] A. Miné. *Weakly Relational Numerical Abstract Domains*. PhD thesis, École Polytechnique, Paris, France, March 2005.

[54] Antoine Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19:31–100, 2006.

[55] John E. Mitchell. Cutting plane algorithms for integer programming. In *Encyclopedia of Optimization*, pages 525–533. Kluwer Academic Publishers, 2001.

[56] R. E. Moore. *Interval Analysis*. Prentice Hall, Englewood Cliffs, NJ, 1966.

[57] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games – Volume II*, number 28 in Annals of Mathematics Studies, pages 51–73. Princeton University Press, Princeton, New Jersey, 1953.

[58] M. Müller-Olm and H. Seidl. A note on Karr's algorithm. In J. Diaz, J. Karhumäki, and A. Lepistö et al., editors, *Automata, Languages and Programming: Proceedings of the 31st International Colloquium (ICALP 2004)*, volume 3142 of *Lecture Notes in Computer Science*, pages 1016–1028, Turku, Finland, 2004. Springer-Verlag, Berlin.

[59] M. Müller-Olm and H. Seidl. Precise interprocedural analysis through linear algebra. In N. D. Jones and X. Leroy, editors, *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2004)*, pages 330–341, Venice, Italy, 2004. ACM Press.

[60] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. In M. Sagiv, editor, *Programming Languages and Systems, Proceedings of the 14th European Symposium on Programming*, volume 3444 of *Lecture Notes in Computer Science*, pages 46–60, Edinburgh, UK, 2005. Springer-Verlag, Berlin.

[61] M. Müller-Olm and H. Seidl. A generic framework for interprocedural analysis of numerical properties. In C. Hankin and I. Siveroni, editors, *Static Analysis: Proceedings of the 12th International Symposium*, volume 3672 of *Lecture Notes in Computer Science*, pages 235–250, London, UK, 2005. Springer-Verlag, Berlin.

[62] M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. *ACM Trans. Program. Lang. Syst.*, 29(5), 2007.

[63] Tsuneo Nakanishi and Akira Fukuda. Modulo interval arithmetic and its application to program analysis. *Information Processing Society of Japan*, 42(4):829–837, 2001.

[64] Tsuneo Nakanishi, Kazuki Joe, Constantine D. Polychronopoulos, and Akira Fukuda. The modulo interval: A simple and practical representation for program analysis. In *IEEE PACT*, pages 91–96, 1999.

[65] G. Nelson and D. C. Oppen. Fast decision algorithms based on Union and Find. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS'77)*, pages 114–119, Providence, RI, USA, 1977. IEEE Computer Society Press. The journal version of this paper is [66].

[66] G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980. An earlier version of this paper is [65].

[67] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1988.

[68] S. P. K. Nookala and T. Risset. A library for Z-polyhedral operations. *Publication interne* 1330, IRISA, Campus de Beaulieu, Rennes, France, 2000.

[69] M. D. Potter. *Sets: An Introduction*. Oxford University Press, 1990.

[70] W. Pugh. A practical algorithm for exact array dependence analysis. *Communications of the ACM*, 35(8):102–114, 1992.

[71] P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra. Technical Report 1016, IRISA, Campus Universitaire de Bealieu, Rennes, France, July 1996.

[72] P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra using a canonic representation. *Parallel Processing Letters*, 7(2):181–194, 1997.

[73] Thomas W. Reps, Gogul Balakrishnan, and Junghee Lim. Intermediate-representation recovery from low-level code. In John Hatcliff and Frank Tip, editors, *PEPM*, pages 100–111. ACM, 2006.

[74] E. Rodríguez-Carbonell and D. Kapur. An abstract interpretation approach for automatic generation of polynomial invariants. In Giacobazzi [35], pages 280–295.

[75] S. Sankaranarayanan, H. Sipma, and Z. Manna. Constraint-based linear-relations analysis. In Giacobazzi [35], pages 53–68.

[76] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1999.

[77] Axel Simon, Andy King, and Jacob M. Howe. Two variables per linear inequality as an abstract domain. In M. Leuschel, editor, *LOPSTR*, volume 2664 of *Lecture Notes in Computer Science*, pages 71–89, Madrid, Spain, 2002. Springer-Verlag, Berlin.

[78] J. Truss. *Discrete Mathematics for Computer Scientists*. Addison-Wesley Longman Limited, 1999.

[79] A. Venet and G. Brat. Precise and efficient static array bound checking for large embedded C programs. In *Proceedings of the ACM SIGPLAN 2004 Conference on Programming Language Design and Implementation (PLDI'04)*, pages 231–242, Washington, DC, USA, 2004. ACM Press.

[80] D. K. Wilde. A library for doing polyhedral operations. Master's thesis, Oregon State University, Corvallis, Oregon, December 1993. Also published as IRISA *Publication interne* 785, Rennes, France, 1993.

[81] L. A. Wolsey. *Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1998.

Appendix A

Declared Publication

A copy of [3] is now given.

Grids: A Domain for Analyzing the Distribution of Numerical Values*

Roberto Bagnara[1], Katy Dobson[2], Patricia M. Hill[2], Matthew Mundell[2], and Enea Zaffanella[1]

[1] Department of Mathematics, University of Parma, Italy,
{bagnara,zaffanella}@cs.unipr.it
[2] School of Computing, University of Leeds, UK,
{katyd,hill,mattm}@comp.leeds.ac.uk

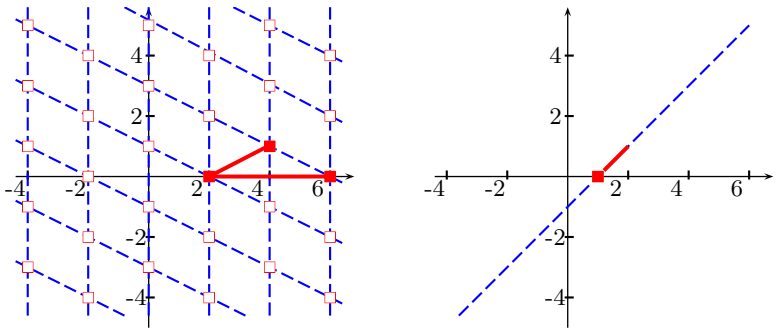
Abstract. This paper explores the abstract domain of *grids*, a domain that is able to represent sets of equally spaced points and hyperplanes over an n-dimensional vector space. Such a domain is useful for the static analysis of the patterns of distribution of the values program variables can take. We present the domain, its representation and the basic operations on grids necessary to define the abstract semantics. We show how the definition of the domain and its operations exploit well-known techniques from linear algebra as well as a dual representation that allows, among other things, for a concise and efficient implementation.

1 Introduction

We distinguish between two kinds of numerical information about the values program variables can take: outer *limits* (or bounds within which the values must lie) and the pattern of *distribution* of these values. Both kinds of information have important applications: in the field of automatic program verification, limit information is crucial to ensure that array accesses are within bounds, while distribution information is what is required to ensure that external memory accesses obey the alignment restriction imposed by the host architecture. In the field of program optimization, limit information can be used to compile out various kinds of run-time tests, whereas distribution information enables several transformations for efficient parallel execution as well as optimizations that enhance cache behavior.

Both limit and distribution information often come in a *relational* form; for instance, the outer limits or the pattern of possible values of one variable may depend on the values of one or more other variables. Domains that can capture relational information are generally much more complex than domains that do not have this capability; in exchange they usually offer significantly more precision, often important for the overall performance of the client application. Relational

* This work has been partly supported by EPSRC project EP/C520726/1 "Numerical Domains for Software Analysis," by MIUR project "AIDA — Abstract Interpretation: Design and Applications," and by a Royal Society (ESEP) award.



(a) The grid $\mathcal{L}$ (b) The grid $\mathcal{L}'$

Fig. 1. Congruence and generator systems representing two grids in $\mathbb{R}^2$

limit information can be captured, among other possibilities, by means of *polyhedral domains*, that is, domains that represent regions of some n-dimensional vector space bounded by a finite set of hyperplanes [10]. Although polyhedral domains such as the domain of convex polyhedra have been thoroughly researched and are widely used, relational domains for representing the (linear) distribution of numerical values have been less well researched. Moreover, as far as we know and at the time of writing, there is no available implementation providing all the basic operations needed by a relational abstract domain for distribution information. This is in spite of the fact that previous research has shown that a knowledge about the (discrete) distribution of numerical information, especially when combined with that of the limit information, can significantly improve the quality of the analysis results [1].

This paper closes this gap by providing a complete account of the relational domain of *grids*; a domain for capturing numerical distribution information. It includes a detailed survey of previous work in this area; gives two representations for the domain; outlines how these can be reduced and also how to convert between them; and shows how this double description directly supports methods for comparing, joining and intersecting elements of this domain. The paper also outlines affine image and preimage operations and two new widenings for grids.

Grids in a Nutshell. Figure 1 illustrates two ways of describing a grid; either by means of a finite set of congruence relations that all grid points must satisfy (given by dashed lines) or by means of a finite set of generating vectors used for constructing the grid points and lines (given by filled squares and thick lines).

The squares in Figure 1(a) illustrate a grid $\mathcal{L}$ indicating possible values of integer variables x and y resulting from executing the program fragment in Figure 2 for any value of `m`. The congruence relations $x = 0 \pmod 2$ and $x + 2y = 2 \pmod 4$ are represented by the vertical dashed lines and sloping lines, respectively. The set of congruence relations $\mathcal{C} = \{x = 0 \pmod 2, x + 2y = 2 \pmod 4\}$, called a *congruence system*, is said to *describe* $\mathcal{L}$. The filled squares mark the points $\boldsymbol{p}_1 = \binom{2}{0}$, $\boldsymbol{p}_2 = \binom{6}{0}$ and $\boldsymbol{p}_3 = \binom{4}{1}$ while all the squares (both filled and unfilled) mark points $\boldsymbol{v} = \pi_1\boldsymbol{p}_1 + \pi_2\boldsymbol{p}_2 + \pi_3\boldsymbol{p}_3$, where $\pi_1, \pi_2, \pi_3 \in \mathbb{Z}$

and $\pi_1 + \pi_2 + \pi_3 = 1$. The set of *points* $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3\}$ is said to *generate* $\mathcal{L}$. Some of these generating points can be replaced by *parameters* that give the gradient and distance between neighboring points. Specifically, by subtracting the point $\boldsymbol{p}_1$ from each of the other two generating points $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$, we obtain the parameters $\boldsymbol{q}_2 = \binom{4}{0}$ and $\boldsymbol{q}_3 = \binom{2}{1}$ for $\mathcal{L}$ that are marked by the thick lines between points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ and points $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$, respectively. It follows that each point $\boldsymbol{v} \in \mathcal{L}$ can be written as $\boldsymbol{v} = \boldsymbol{p}_1 + \pi_2\boldsymbol{q}_2 + \pi_3\boldsymbol{q}_3$ for some $\pi_2, \pi_3 \in \mathbb{Z}$.

The dashed line in Figure 1(b) illustrates the grid $\mathcal{L}'$ defining the line $x = y + 1$ and marks the vectors of values of the real variables x and y after an assignment `x := y + 1`, assuming that nothing is known about the value of y. As equalities are congruences modulo 0, the set $\mathcal{C}' = \{x - y = 1\}$ is also called a congruence system and describes $\mathcal{L}'$. Observe that the grid $\mathcal{L}'$ consists of all points that can be obtained as $\lambda\boldsymbol{\ell} + \boldsymbol{p}'$, for any $\lambda \in \mathbb{R}$, where $\boldsymbol{\ell} = \binom{1}{1}$ and $\boldsymbol{p}' = \binom{1}{0}$; the vector $\boldsymbol{\ell}$, called a *line*, defines a gradient and the vector $\boldsymbol{p}'$ is a generating point marking a position for the line (illustrated in Figure 1(b) by the thick line and the filled square, respectively).

```
x := 2; y := 0;   (P1)
for i := 1 to m   (P2)
  if ... then
    x := x + 4    (P3)
  else
    x := x + 2;
    y := y + 1    (P4)
  endif           (P5)
endfor
```

Fig. 2. Fragment based on an example in [10]

From what we have just seen, any grid can be represented both by a congruence system and by a *generator system*. The latter may consist of three components: a set of lines, a set of parameters and a set of points. For instance, the triples $\mathcal{G}_1 = \big(\varnothing, \varnothing, P\big)$ and $\mathcal{G}_2 = \big(\varnothing, \{\boldsymbol{q}_2, \boldsymbol{q}_3\}, \{\boldsymbol{p}_1\}\big)$ are both generator systems for $\mathcal{L}$ while the triple $\mathcal{G}' = \big(\{\boldsymbol{\ell}\}, \varnothing, \{\boldsymbol{p}'\}\big)$ is a generator system for $\mathcal{L}'$.

Contributions. The paper provides an account of the relational domain of *grids*, fully implemented within the Parma Polyhedra Library [2, 4]. In this section we provide the first comprehensive survey of the main research threads concerning these and similar domains. The other contributions are given below.

Minimizing representations. Assuming the grid is represented by a congruence and generator system in an n-dimensional vector space consisting of m congruences or generators, then we outline algorithms for minimizing the representation (based on the Hermite normal form algorithm [29]) that have worst-case complexity $\mathrm{O}\big(n^2m\big)$. Note that previous proposals for minimization such as those in [14, 23] have worse complexity bounds (see below).

Converting representations. The congruence and generator representations described informally above form the two components of a double description method for the grid domain very similar to that for convex polyhedra [20]. For a double description method, conversion algorithms between the two systems are needed; we show how conversion can be implemented using any matrix inversion algorithm, inheriting the corresponding worst-case complexity. For instance, the

complexity is $O(n^3)$ when adopting the standard Gaussian elimination method; since matrix inversion has the same worst-case complexity as matrix multiplication, better theoretical complexity bounds apply [5]. Previous proposals for congruence to generator conversion have complexity no better than $O(n^4)$ [15].

Grid operations. For static analysis, it is useful to provide all the set-theoretic lattice operations for grids (assuming the usual subset ordering) such as comparison, join and meet. We show that these operations are straightforward given the availability of the appropriate representation(s) in minimal form; and hence show that some have complexities strictly better than that of previous proposals [14]. We also describe a grid difference operator which is new to this paper.

Affine transformation operators. Affine image and preimage operators can be used to capture the effect of assignment statements in a program when the expression is linear although, as noted by Müller-Olm and Seidl in [21], analyses that use affine spaces for approximating the semantics of procedures are not sufficiently precise to detect all valid affine relations for programs with procedures. Here we specify, for the domain of grids, the affine image and preimage operators for a *single update* where only one dimension is modified.

Widenings. It was observed by Granger [15], that, if the grid generators can be in the rationals, then the grid domain does not satisfy the ascending chain condition; so, to guarantee termination of the analysis, a widening operation is required. In [15, Proposition 10], a widening is given for non-relational grids that returns a line parallel to an axis whenever the modulus for that dimension changes. It is then proposed that a generalized form of this could be used as a widening for relational grids; however, exactly how this is to be done is unclear. In this paper, we define two possible generalizations which come with simple syntactic checks that have efficient implementations.

Related Work. In [12], Granger shows how a static analysis can usefully employ a simple *non-relational* grid domain (that is a grid described by congruences of the form $x = c \pmod f$ where c and f are integers) and that this domain can obtain more precise information for applications such as automatic vectorization. Larsen et al. [17] also developed a static analyzer over a non-relational grid domain specifically designed to detect when dynamic memory addresses are congruent with respect to a given modulus; they show that, this information helps in the construction of a comprehensive set of program transformations for saving energy on low-power architectures and improving performance on multimedia processors. We note that these applications should carry over to the more complex domain considered here. In addition, Miné has shown how to construct, from the non-relational congruence domain in [12], a zone-congruence domain (that is, a domain that only allows *weakly relational* congruences that have the form $x - y = a \pmod b$ where a and b are rationals) [19].

Concerning *fully relational* domains, note that the use of a domain of linear *equality* relations for program analysis had already been studied by Karr [16].

In [14], Granger generalized this to provide a domain of linear *congruence* relations on an integral domain, i.e., a domain generated by integral vectors in n-dimensions; and then, in [13, 15], generalizes the results to the full grid domain. In [13–15], domain elements are represented by congruence and generator systems similar to the ones defined here. Standard algorithms for solving linear equations are used in converting from generator to congruence systems; however, a more complex $\mathrm{O}(n^4)$ algorithm is provided for converting from congruence to generator systems. Assuming the number of generators is $n + 1$, the algorithm for minimizing the generator system has complexity $\mathrm{O}(n^3 \log_2 n)$. Operators for comparing grids and computing the greatest lower and least upper bounds are also described. In particular, the join operation defined in [14] has complexity $\mathrm{O}(n^4 \log_2 n)$, since the generators of one grid are added, one at a time, to the generators of the other; after each addition the minimization algorithm is applied to compute a new linearly independent set. The grid meet operation which also minimizes the addition of one congruence at a time has complexity $\mathrm{O}(n^4)$.

The problem of how best to *apply* the grid domain in a program analyzer, has been studied by Müller-Olm and Seidl in [23] also building on the work of Karr [16]. Here, the prime focus is for the design of an *interprocedural* analysis for programs containing assignment statements and procedure calls. The algorithm has three stages: first, for each program point, a matrix M containing a (minimized) set of generators (i.e., vectors of values that hold at that point) is found; secondly, the determinant f of M is computed; thirdly, a congruence system with modulo f that satisfies all the vectors in M is determined. Stage one is similar to that proposed by Granger [14] for minimizing a set of generators. Stages two and three differ from the conversion in [14] in that the modulus f is computed separately and used to reduce the sizes of the coordinates. Note that the framework described in [23] subsumes previous works by the same authors.

Following an independent stream of research, Ancourt [1] considered the domain of $\mathbb{Z}$-*polyhedra*; that is a domain of *integral lattices* intersected with the domain of convex polyhedra (see also [24–26]). We are primarily interested here in the "integral lattices" component which may be seen as a subdomain of the domain of grids where the grid is full dimensional and all the grid points are integral vectors. The representation of these integral lattices is a special case of our generator representation where, for n dimensions, there must be exactly one point and n linearly independent parameters, all of which must be integral. There is no support for a congruence representation.

All the operations on $\mathbb{Z}$-polyhedra (and therefore the lattices) require canonic representations; hence Quinton et al. [25, 26] define a canonical form for these lattices with a method for its computation. We note that the algorithm for computing the canonic form has complexity $\mathrm{O}(n^4)$, where n is the number of dimensions of the vector space. Other operations provided are those of lattice intersection, affine image and affine preimage. As there is no congruence representation, the intersection of two lattices is computed directly from the generator representations [1]; a refined version of this method is provided in [25] which we note that, as for computing the canonic form, has complexity $\mathrm{O}(n^4)$. The opera-

tions of grid join and grid difference (as defined here) are not considered; instead the union operator takes two lattices $\mathcal{L}_1$ and $\mathcal{L}_2$ and returns the set $\{\mathcal{L}_1, \mathcal{L}_2\}$ unless one (say $\mathcal{L}_1$) is contained in the other, in which case they return the larger, $\mathcal{L}_2$. Similarly the difference operation returns a set of lattices representing the set difference $\mathcal{L}_1 \setminus \mathcal{L}_2$. The domain of integral lattices has been implemented in PolyLib [18] following the approach in [25, 26]. This means that only the generator representation is supported and some operations return *sets* of lattices while others manipulate and simplify these sets.

The *homogeneous form* of a representation given in Section 4, is required by the conversion algorithm. This form is not new to this paper; in fact several researchers have observed this. For instance, Granger [14] describes a map from a linear congruence system in n variables to a homogeneous one in $n+1$ variables; Nookala and Risset [24] explain that the PolyLib [18] adds a dimension to make the (generator) representation homogeneous; while Müller-Olm and Seidl [23] consider *extended states* where vectors have an extra 0'th component.

Plan of the Paper. Preliminary concepts and notation are given in Section 2. Section 3 introduces a grid together with its congruence and generator representations while Section 4 provides the main algorithms needed to support the double description. Section 5 introduces grid widening and the paper concludes in Section 6. A long version of the paper containing all proofs is available at `http://www.comp.leeds.ac.uk/hill/Papers/papers.html`.

2 Preliminaries

The *cardinality* of a set S is denoted by $\# S$. The set of integers is denoted by $\mathbb{Z}$, rationals by $\mathbb{Q}$ and reals by $\mathbb{R}$. The complexities will assume a unit cost for every arithmetic operation.

Matrices and Vectors. If H is a matrix in $\mathbb{R}^{n\times m}$, the *transposition* of H is denoted by $H^{\mathrm{T}} \in \mathbb{R}^{m\times n}$. A vector $\boldsymbol{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$ is also regarded as a matrix in $\mathbb{R}^{n\times 1}$. The *scalar product* of vectors $\boldsymbol{v}$ and $\boldsymbol{w} \in \mathbb{R}^n$, denoted by $\langle \boldsymbol{v}, \boldsymbol{w}\rangle$, is the real number $\boldsymbol{v}^{\mathrm{T}}\boldsymbol{w} = \sum_{i=1}^{n} v_i w_i$. The vector $\boldsymbol{e}_i \in \mathbb{R}^n$ has 1 in the i-th position and 0 in every other position. We let

$$\mathrm{piv}_{<}(\boldsymbol{v}) := \begin{cases} 0 & \text{if } \boldsymbol{v} = 0 \\ \max\{i \mid 1 \le i \le n, v_i \neq 0\} & \text{if } \boldsymbol{v} \neq \boldsymbol{0} \end{cases}$$

$$\mathrm{piv}_{>}(\boldsymbol{v}) := \begin{cases} n+1 & \text{if } \boldsymbol{v} = 0 \\ \min\{i \mid 1 \le i \le n, v_i \neq 0\} & \text{if } \boldsymbol{v} \neq \boldsymbol{0}. \end{cases}$$

We write $\boldsymbol{v} \Uparrow \boldsymbol{v}'$, if $\mathrm{piv}_{<}(\boldsymbol{v}) = \mathrm{piv}_{<}(\boldsymbol{v}') = k$ and either $k = 0$ or $v_k = v'_k$ and $\boldsymbol{v} \Downarrow \boldsymbol{v}'$, if $\mathrm{piv}_{>}(\boldsymbol{v}) = \mathrm{piv}_{>}(\boldsymbol{v}') = k$ and either $k = n+1$ or $v_k = v'_k$.

Integer Combinations. The set $S = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\} \subseteq \mathbb{R}^n$ is *affinely independent* if, for all $\boldsymbol{\lambda} \in \mathbb{R}^k$, $\boldsymbol{\lambda} = \mathbf{0}$ is the only solution of $\{\sum_{i=1}^k \lambda_i \boldsymbol{v}_i = \mathbf{0},\ \sum_{i=1}^k \lambda_i = 0\}$. For all $\boldsymbol{\lambda} \in \mathbb{R}^k$, the vector $\boldsymbol{v} = \sum_{j=1}^k \lambda_j \boldsymbol{v}_j$ is said to be a *linear* combination of S. This combination is *affine*, if $\sum_{j=1}^k \lambda_j = 1$; and *integral*, if $\boldsymbol{\lambda} \in \mathbb{Z}^k$. The set of all linear (resp., affine, integral, integral and affine) combinations of S is denoted by linear.hull (resp., affine.hull(S), int.hull(S), int.affine.hull(S)).

Congruences and Congruence Relations. For any $a, b, f \in \mathbb{R}$, $a \equiv_f b$ denotes the *congruence* $\exists \mu \in \mathbb{Z}\ .\ a - b = \mu f$. Let $\mathbb{S} \in \{\mathbb{Q}, \mathbb{R}\}$. For each vector $\boldsymbol{a} \in \mathbb{S}^n$ and scalars $b, f \in \mathbb{S}$, the notation $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b$ stands for the *linear congruence relation in* $\mathbb{S}^n$ defined by the set $\{\ \boldsymbol{v} \in \mathbb{R}^n \mid \exists \mu \in \mathbb{Z}\ .\ \langle \boldsymbol{a}, \boldsymbol{v} \rangle = b + \mu f\ \}$; when $f \neq 0$, the relation is said to be *proper*; $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_0 b$ denotes the equality $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b$. Thus, provided $\boldsymbol{a} \neq \mathbf{0}$, the relation $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b$ defines the set of affine hyperplanes $\{\ (\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b + \mu f) \mid \mu \in \mathbb{Z}\ \}$; when $\boldsymbol{a} = \mathbf{0}$, we assume that $b \neq 0$; if $b \equiv_f 0$, $\langle \mathbf{0}, \boldsymbol{x} \rangle \equiv_f b$ defines the universe $\mathbb{R}^n$ and the empty set, otherwise.

Any vector that satisfies $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b + \mu f$ for some $\mu \in \mathbb{Z}$ is said to *satisfy* the relation $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b$. Congruence relations in $\mathbb{S}^n$, such as $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_1 b$ and $\langle 2\boldsymbol{a}, \boldsymbol{x} \rangle \equiv_2 2b$, defining the same hyperplanes are considered equivalent.

The pivot notation for vectors is extended to congruences: if $\beta = (\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f a_0)$ then $\mathrm{piv}_<(\beta) := \mathrm{piv}_<(\boldsymbol{a})$; if $\gamma = (\langle \boldsymbol{c}, \boldsymbol{x} \rangle \equiv_g c_0)$ and $g\boldsymbol{a} \Uparrow f\boldsymbol{c}$, then we write $\beta \Uparrow \gamma$; so that β and γ are either both equalities or both proper congruences.

3 The Grid Domain

Here we introduce grids and their representation. Note that the use of the word 'grid' here is to avoid confusion with the meaning of 'lattice' (used previously for elements similar to a grid) in its set-theoretic context (particularly relevant when working in abstract interpretation).

Grids and the Congruence Representation. A *congruence system in* $\mathbb{Q}^n$ is a finite set of congruence relations $\mathcal{C}$ in $\mathbb{Q}^n$. As we do not distinguish between syntactically different congruences defining the same set of vectors, we can assume that all proper congruences in $\mathcal{C}$ have modulus 1.

Definition 1. *Let $\mathcal{C}$ be a congruence system in $\mathbb{R}^n$. If $\mathcal{L}$ is the set of vectors in $\mathbb{R}^n$ that satisfy all the congruences in $\mathcal{C}$, we say that $\mathcal{L}$ is a* grid described by *a congruence system $\mathcal{C}$ in $\mathbb{Q}^n$. We also say that $\mathcal{C}$* is a congruence system for *$\mathcal{L}$ and write $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$. If $\mathrm{gcon}(\mathcal{C}) = \varnothing$, then we say that $\mathcal{C}$ is* inconsistent*.*

The grid domain *$\mathbb{G}_n$ is the set of all grids in $\mathbb{R}^n$ ordered by the set inclusion relation, so that $\varnothing$ and $\mathbb{R}^n$ are the bottom and top elements of $\mathbb{G}_n$ respectively.*

The vector space $\mathbb{R}^n$ is called the *universe* grid. In set theoretical terms, $\mathbb{G}_n$ is a *lattice* under set inclusion. Many algorithms given here will require the congruence systems not only to have minimal cardinality but also such that the coefficients of (a permutation of) the congruences can form a triangular matrix.

Definition 2. *Suppose $\mathcal{C}$ is a congruence system in $\mathbb{Q}^n$. Then we say that $\mathcal{C}$ is in* minimal form *if either $\mathcal{C} = \{\langle \mathbf{0}, \boldsymbol{x}\rangle \equiv_0 1\}$ or $\mathcal{C}$ is consistent and, for each congruence $\beta = \big(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\big) \in \mathcal{C}$, the following hold:*

1. *if $\mathrm{piv}_<(\beta) = k$, then $k > 0$ and $a_k > 0$;*
2. *for all $\beta' \in \mathcal{C} \setminus \{\beta\}$, $\mathrm{piv}_<(\boldsymbol{\beta}') \neq \mathrm{piv}_<(\boldsymbol{\beta})$.*

Proposition 1. *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$ and $m = \#\,\mathcal{C}$. Then there exists an algorithm for finding a congruence system $\mathcal{C}'$ in minimal form with worst-case complexity $\mathrm{O}\big(n^2 m\big)$ such that $\mathrm{gcon}(\mathcal{C}) = \mathrm{gcon}(\mathcal{C}')$.*

Note that the algorithm mentioned in Proposition 1, is based on the Hermite normal form algorithm; details about the actual algorithm are given in the proof. Note also, that when $m < n$, the complexity of this algorithm is just $\mathrm{O}\big(m^2 n\big)$.

The Generator Representation. Let $\mathcal{L}$ be a grid in $\mathbb{G}_n$. Then

- a vector $\boldsymbol{p} \in \mathcal{L}$ is called a *point* of $\mathcal{L}$;
- a vector $\boldsymbol{q} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is called a *parameter* of $\mathcal{L}$ if $\mathcal{L} \neq \varnothing$ and $\boldsymbol{p} + \mu\boldsymbol{q} \in \mathcal{L}$, for all points $\boldsymbol{p} \in \mathcal{L}$ and all $\mu \in \mathbb{Z}$;
- a vector $\boldsymbol{\ell} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ is called a *line* of $\mathcal{L}$ if $\mathcal{L} \neq \varnothing$ and $\boldsymbol{p} + \lambda\boldsymbol{\ell} \in \mathcal{L}$, for all points $\boldsymbol{p} \in \mathcal{L}$ and all $\lambda \in \mathbb{R}$.

If L, Q and P are finite sets of vecors in $\mathbb{R}^n$ and

$$\mathcal{L} := \mathrm{linear.hull}(L) + \mathrm{int.hull}(Q) + \mathrm{int.affine.hull}(P)$$

where the symbol '+' denotes the Minkowski's sum,[3] then $\mathcal{L} \in \mathbb{G}_n$ is a grid (see [29, Section 4.4] and also Proposition 7). The 3-tuple (L, Q, P), where L, Q and P denote sets of lines, parameters and points, respectively, is said to be a *generator system* in $\mathbb{Q}^n$ for $\mathcal{L}$ and we write $\mathcal{L} = \mathrm{ggen}\big((L, Q, P)\big)$. Note that, for any grid $\mathcal{L}$ in $\mathbb{G}_n$, there is a generator system (L, Q, P) in $\mathbb{Q}^n$ for $\mathcal{L}$ (see again [29, Section 4.4] and also Proposition 6). Note also that the grid $\mathcal{L} = \mathrm{ggen}\big((L, Q, P)\big) = \varnothing$ if and only if the set of points $P = \varnothing$. If $P \neq \varnothing$, then $\mathcal{L} = \mathrm{ggen}\big((L, \varnothing, Q_{\boldsymbol{p}} \cup P)\big)$ where, for some $\boldsymbol{p} \in P$, $Q_{\boldsymbol{p}} = \{\, \boldsymbol{p} + \boldsymbol{q} \in \mathbb{R}^n \mid \boldsymbol{q} \in Q \,\}$.

As for congruence systems, for many procedures in the implementation, it is useful if the generator systems have a minimal number of elements.

Definition 3. *Suppose $\mathcal{G} = (L, Q, P)$ is a generator system in $\mathbb{Q}^n$. Then we say that $\mathcal{G}$ is in* minimal form *if either $L = Q = P = \varnothing$ or $\#\,P = 1$ and, for each generator $\boldsymbol{v} \in L \cup Q$, the following hold:*

1. *if $\mathrm{piv}_>(\boldsymbol{v}) = k$, then $v_k > 0$;*
2. *for all $\boldsymbol{v}' \in (L \cup Q) \setminus \{\boldsymbol{v}\}$, $\mathrm{piv}_>(\boldsymbol{v}') \neq \mathrm{piv}_>(\boldsymbol{v})$.*

Proposition 2. *Let $\mathcal{G} = (L, Q, P)$ be a generator system in $\mathbb{Q}^n$ and $m = \#\,L + \#\,Q + \#\,P$. Then there exists an algorithm for finding a generator system $\mathcal{G}'$ in minimal form with worst-case complexity $\mathrm{O}\big(n^2 m\big)$ such that $\mathrm{ggen}(\mathcal{G}') = \mathrm{ggen}(\mathcal{G})$.*

As for Proposition 1, the algorithm mentioned in Proposition 2 is based on the Hermite normal form algorithm. Note also that, when $m < n$, the complexity of this algorithm is again just $\mathrm{O}\big(m^2 n\big)$.

[3] This is defined, for each $S, T \subseteq \mathbb{R}^n$, by $S + T := \{\, \boldsymbol{s} + \boldsymbol{t} \in \mathbb{R}^n \mid \boldsymbol{s} \in S, \boldsymbol{t} \in T \,\}$.

Double Description. We have shown that any grid $\mathcal{L}$ can be described by using a congruence system $\mathcal{C}$ and also generated by a generator system $\mathcal{G}$. For the same reasons as for the polyhedral domain, it is useful to represent the grid $\mathcal{L}$ by the *double description* $(\mathcal{C}, \mathcal{G})$. Just as for the double description method for convex polyhedra, in order to maintain and exploit such a view of a grid, an implementation must include algorithms for converting a representation of one kind into a representation of the other kind and for minimizing both representations. Note that having easy access to both representations is assumed in the implementation of many grid operators including those described here.

Suppose we have a double description $\big(\mathcal{C}, \mathcal{G}\big)$ of a grid $\mathcal{L} \in \mathbb{G}_n$, where both $\mathcal{C}$ and $\mathcal{G}$ are in minimal form. Then, it follows from the definition of minimal form that $\#\,\mathcal{C} \leq n+1$ and $\#\,L + \#\,Q \leq n$. In fact, we have a stronger result.

Proposition 3. *Let $(\mathcal{C}, \mathcal{G})$ be a double description where both $\mathcal{C}$ and $\mathcal{G}$ are in minimal form. Letting $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$, where $\mathcal{E}$ and $\mathcal{F}$ are sets of equalities and proper congruences, respectively, and $\mathcal{G} = (L, Q, P)$, then $\#\,\mathcal{F} = \#\,Q = n - \#\,L - \#\,\mathcal{E}$.*

Example 1. Consider the grids $\mathcal{L}$ and $\mathcal{L}'$ in Figure 1. The congruence systems $\mathcal{C}$ and $\mathcal{C}'$ are in minimal form and the generator systems $\mathcal{G}_2$ and $\mathcal{G}'$ are also in minimal form; however, $\mathcal{G}_1$ is not in minimal form as it contains more than one point. Furthermore, for $i = 1, 2$, the pairs $(\mathcal{C}, \mathcal{G}_i)$ are double descriptions for $\mathcal{L}$ while $(\mathcal{C}', \mathcal{G}')$ is a double description for $\mathcal{L}'$.

Comparing Grids. For any pair of grids $\mathcal{L}_1 = \mathrm{ggen}\big((L, Q, P)\big)$, $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C})$ in $\mathbb{G}_n$, we can decide whether $\mathcal{L}_1 \subseteq \mathcal{L}_2$ by checking if every generator in (L, Q, P) satisfies every congruence in $\mathcal{C}$. Note that a parameter or line $\boldsymbol{v}$ *satisfies* a congruence $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_f b$ if $\langle \boldsymbol{a}, \boldsymbol{v} \rangle \equiv_f 0$. Therefore, assuming the systems $\mathcal{C}$ and $\mathcal{G}$ are already in minimal form, the complexity of comparison is $\mathrm{O}\big(n^3\big)$.

Given that it is known that one grid is a subset of another, there are quicker tests for checking equality - the following definition is used in their specification.

Definition 4. *Let $\mathcal{C}_1, \mathcal{C}_2$ be congruence systems in minimal form. Then $\mathcal{C}_1, \mathcal{C}_2$ are said to be* pivot equivalent *if, for each $i, j \in \{1, 2\}$ where $i \neq j$, for each $\beta \in \mathcal{C}_i$, there exists $\gamma \in \mathcal{C}_j$ such that $\beta \Uparrow \gamma$.*

Let $\mathcal{G}_1 = \big(L_1, Q_1, \{\boldsymbol{p}_1\}\big)$ and $\mathcal{G}_2 = \big(L_2, Q_2, \{\boldsymbol{p}_2\}\big)$ be generator systems in minimal form. Then $\mathcal{G}_1, \mathcal{G}_2$ are said to be pivot equivalent *if, for each $i, j \in \{1, 2\}$ where $i \neq j$: for each $\boldsymbol{q}_i \in Q_i$, there exists $\boldsymbol{q}_j \in Q_j$ such that $\boldsymbol{q}_i \Downarrow \boldsymbol{q}_j$; and, for each $\boldsymbol{\ell}_i \in L_i$, there exists $\boldsymbol{\ell}_j \in L_j$ such that $\mathrm{piv}_{>}(\boldsymbol{\ell}_i) = \mathrm{piv}_{>}(\boldsymbol{\ell}_j)$.*

Proposition 4. *Let $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1) = \mathrm{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2) = \mathrm{ggen}(\mathcal{G}_2)$ be non-empty grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$. If $\mathcal{C}_1$ and $\mathcal{C}_2$ are pivot equivalent congruence systems in minimal form or $\mathcal{G}_1$ and $\mathcal{G}_2$ are pivot equivalent generator systems in minimal form, then $\mathcal{L}_1 = \mathcal{L}_2$.*

It follows from Proposition 4, that provided $\mathcal{L}_1 \subseteq \mathcal{L}_2$ and $\mathcal{L}_1$ and $\mathcal{L}_2$ have both their generator or congruence systems already in minimal form, then the complexity of checking if $\mathcal{L}_1 = \mathcal{L}_2$ is just $\mathrm{O}\big(n\big)$. Moreover, if it is found that one pair of corresponding pivot elements of the congruence or generator systems differ, then we can immediately deduce that the grids they describe also differ.

Intersection and Grid Join. For grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *intersection* of $\mathcal{L}_1$ and $\mathcal{L}_2$, defined as the set intersection $\mathcal{L}_1 \cap \mathcal{L}_2$, is the largest grid included in both $\mathcal{L}_1$ and $\mathcal{L}_2$; similarly, the *grid join* of $\mathcal{L}_1$ and $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \oplus \mathcal{L}_2$, is the smallest grid that includes both $\mathcal{L}_1$ and $\mathcal{L}_2$. In theoretical terms, the intersection and grid join operators are the binary *meet* and *join* operators on the lattice $\mathbb{G}_n$. They can easily be computed; if $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1) = \operatorname{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2) = \operatorname{ggen}(\mathcal{G}_2)$, then $\mathcal{L}_1 \cap \mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_1 \cup \mathcal{C}_2)$ and $\mathcal{L}_1 \oplus \mathcal{L}_2 = \operatorname{ggen}(\mathcal{G}_1 \cup \mathcal{G}_2)$.

In practice, the cost of computing the grid intersection and join depends on a number of factors: if generator systems $\mathcal{G}_1$ and $\mathcal{G}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$ are known, then the complexity of computing $\mathcal{L}_1 \oplus \mathcal{L}_2$ is linear in either $\#\,\mathcal{G}_1$ or $\#\,\mathcal{G}_2$; if, however, only congruence systems $\mathcal{C}_1$ and $\mathcal{C}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$ (not necessarily in minimal form) are known, then the complexity is that of minimizing and converting them which is, at worst, $\mathrm{O}\big(n^2 \max(\#\,\mathcal{C}_1, \#\,\mathcal{C}_2, n)\big)$. A similar argument applies to the complexities of the meet operation. However, the above operations are not directly comparable with the meet and join operations given in [14]. For such a comparison, for instance for the join operation, we assume that generator systems for $\mathcal{L}_1$ and $\mathcal{L}_2$ in minimal form are available (i.e., each with at most $n+1$ generators) and the operation returns a generator system in minimal form for $\mathcal{L}_1 \oplus \mathcal{L}_2$. Then the complexity is $\mathrm{O}\big(n^3\big)$, the complexity of minimizing a generator system with at most $2n+2$ generators, which is strictly better than $\mathrm{O}\big(n^4 \log_2 n\big)$, the complexity of the equivalent operation in [14].

Example 2. Consider the grids $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2)$ in $\mathbb{G}_2$ where $\mathcal{C}_1 := \{x \equiv_2 0,\ -x + y \equiv_3 0\}$ and $\mathcal{C}_2 := \{x \equiv_4 0,\ -x + 2y \equiv_6 0\}$. Then the grid intersection is $\mathcal{L}_1 \cap \mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_1 \cup \mathcal{C}_2)$; thus, as $\mathcal{C} = \{x \equiv_{12} 0,\ y \equiv_3 0\}$ is a reduced form of $\mathcal{C}_1 \cup \mathcal{C}_2$, we have $\mathcal{L}_1 \cap \mathcal{L}_2 = \operatorname{gcon}(\mathcal{C})$.

Consider $\mathcal{L}_1 = \operatorname{ggen}\big((\varnothing, \varnothing, P_1)\big)$ and $\mathcal{L}_2 = \operatorname{ggen}\big((\varnothing, \varnothing, P_2)\big)$ in $\mathbb{G}_2$, where $P_1 := \left(\begin{smallmatrix} 2 & 0 & 0 \\ 2 & 3 & 0 \end{smallmatrix}\right)$ and $P_2 := \left(\begin{smallmatrix} 4 & 0 & 0 \\ 2 & 3 & 0 \end{smallmatrix}\right)$. Then the grid join $\mathcal{L}_1 \oplus \mathcal{L}_2$ is generated by $(\varnothing, \varnothing, P_1 \cup P_2)$; thus, the generator system $\mathcal{G} := \big(\varnothing, \left(\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right)\big)$ is a minimal form of $(\varnothing, \varnothing, P_1 \cup P_2)$ and $\mathcal{L}_1 \oplus \mathcal{L}_2 = \operatorname{ggen}(\mathcal{G})$. Note that here $\mathcal{L}_1 \oplus \mathcal{L}_2 \neq \mathcal{L}_1 \cup \mathcal{L}_2$.

Grid Difference. For grids $\mathcal{L}_1, \mathcal{L}_2 \in \mathbb{G}_n$, the *grid difference* $\mathcal{L}_1 \ominus \mathcal{L}_2$ of $\mathcal{L}_1$ and $\mathcal{L}_2$ is the smallest grid containing the set-theoretic difference of $\mathcal{L}_1$ and $\mathcal{L}_2$.

Proposition 5. *The grid $\mathcal{L}_1 \ominus \mathcal{L}_2$ is returned by the algorithm in Figure 3.*

Assuming $\mathcal{C}_1$ and $\mathcal{C}_2$ are available and in minimal form, it follows from the complexities of minimization, conversion and comparison operations that the grid difference algorithm in Figure 3 has worst-case complexity $\mathrm{O}\big(n^4\big)$.

Affine Images and Preimages. Affine transformations for the vector space $\mathbb{R}^n$ will map hyperplanes to hyperplanes and preserve intersection properties between hyperplanes; such transformations can be represented by matrices in $\mathbb{R}^{n \times n}$. It follows that the set $\mathbb{G}_n$ is closed under the set of all affine transformations for $\mathbb{R}^n$. Simple and useful linear affine transformations for numerical

Input: Nonempty grids $\mathcal{L}_1 = \mathrm{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \mathrm{gcon}(\mathcal{C}_2)$ in $\mathbb{G}_n$.
Output: A grid in $\mathbb{G}_n$.

```
(1)   L' := ∅
(2)   while ∃β = (e ≡_f 0) ∈ C_2
(3)     C_2 := C_2 \ {β}
(4)     if L_1 ⊈ gcon({β})
(5)       if L_1 ⊆ gcon({2e ≡_f 0})
(6)         L_β := gcon(C_1 ∪ {2e − f ≡_2f 0})
(7)         L' := L' ⊕ L_β
(8)       else
(9)         return L_1
(10)  return L'
```

Fig. 3. The grid difference algorithm

domains, including the grids, are provided by the 'single update' affine image and affine preimage operators.

Given a grid $\mathcal{L} \in \mathbb{G}_n$, a variable x_k and linear expression $e = \langle \boldsymbol{a}, \boldsymbol{x} \rangle + b$ with coefficients in $\mathbb{Q}$, the *affine image operator* $\phi(\mathcal{L}, x_k, e)$ maps the grid $\mathcal{L}$ to

$$\Big\{ \big(p_1, \ldots, p_{k-1}, \langle \boldsymbol{a}, \boldsymbol{p} \rangle + b, p_{k+1}, \ldots, p_n\big)^{\mathrm{T}} \in \mathbb{R}^n \;\Big|\; \boldsymbol{p} \in \mathcal{L} \Big\}.$$

Conversely, the *affine preimage operator* $\phi^{-1}(\mathcal{L}, x_k, e)$ maps the grid $\mathcal{L}$ to

$$\Big\{ \boldsymbol{p} \in \mathbb{R}^n \;\Big|\; \big(p_1, \ldots, p_{k-1}, \langle \boldsymbol{a}, \boldsymbol{p} \rangle + b, p_{k+1}, \ldots, p_n\big)^{\mathrm{T}} \in \mathcal{L} \Big\}.$$

Observe that the affine image $\phi(\mathcal{L}, x_k, e)$ and preimage $\phi^{-1}(\mathcal{L}, x_k, e)$ are invertible if and only if the coefficient a_k in the vector $\boldsymbol{a}$ is non-zero.

Program Analysis Using Grids. We show how the grid domain can be used to find properties of the program variables not found using the polyhedra domain [10], constraint-based analysis [28] or polynomial invariants [27].

Example 3. The program fragment in Figure 2 is annotated with program points $\mathtt{P}j$, for $j = 1, \ldots, 5$. Let $\mathcal{L}_j^i \in \mathbb{G}_2$ denote the grid computed at the i-th iteration executed by the point $\mathtt{P}j$. Initially, $\mathcal{L}_j^0 = \varnothing = \mathrm{gcon}\big(\{1 = 0\}\big)$, for $j = 1, \ldots, 5$. After one and two iterations of the loop we have:

$$\begin{aligned}
&\mathcal{L}_1^1 = \mathrm{gcon}\big(\{x = 2,\ y = 0\}\big), \quad \mathcal{L}_2^1 = \mathrm{gcon}\big(\{x = 2,\ y = 0\}\big),\\
&\mathcal{L}_3^1 = \mathrm{gcon}\big(\{x = 6,\ y = 0\}\big), \quad \mathcal{L}_4^1 = \mathrm{gcon}\big(\{x = 4,\ y = 1\}\big),\\
&\mathcal{L}_5^1 = \mathrm{gcon}\big(\{x = 4,\ y = 1\}\big) \oplus \mathrm{gcon}\big(\{x = 6,\ y = 0\}\big)\\
&\quad = \mathrm{gcon}\big(\{x + 2y = 6,\ x \equiv_2 0\}\big),\\
&\mathcal{L}_2^2 = \mathrm{gcon}\big(\{x = 2, y = 0\}\big) \oplus \mathrm{gcon}\big(\{x + 2y = 6, x \equiv_2 0\}\big)\\
&\quad = \mathrm{gcon}\big(\{x + 2y \equiv_4 2, x \equiv_2 0\}\big).
\end{aligned}$$

Subsequent computation steps show that an invariant for `P2` has already been computed since $\mathcal{L}_3^2 = \mathcal{L}_3^1$, $\mathcal{L}_4^2 = \mathcal{L}_4^1$, $\mathcal{L}_5^2 = \mathcal{L}_5^1$ so that $\mathcal{L}_2^3 = \mathcal{L}_2^2$. Thus at the end of the program, the congruences $x + 2y \equiv_4 2$ and $x \equiv_2 0$ hold.

Observe that, using convex polyhedra, a similar analysis will find instead that the inequalities $x - 2y \geq 2$, $x + 2y \geq 6$ and $y \geq 0$ hold [10].

4 Implementation

In this section, we describe convenient internal representations of the congruence and generator systems in terms of arrays (i.e., matrices) and show how matrix inversion provides a basis for converting between these representations.

Homogeneous Representations. A congruence system $\mathcal{C}$ is *homogeneous* if, for all $\big(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\big) \in \mathcal{C}$, we have $b = 0$. Similarly, a generator system (L, Q, P) is *homogeneous* if $\mathbf{0} \in P$. For the implementation, it is convenient to work with a homogeneous system. Thus we first convert any congruence or generator system in $\mathbb{Q}^n$ to a homogeneous system in $\mathbb{Q}^{n+1}$. The extra dimension is denoted with a 0 subscript; the vector $\hat{\boldsymbol{x}} = (x_0, \ldots, x_n)^{\mathrm{T}}$; and $\boldsymbol{e}_0$ denotes the vector $(1, \mathbf{0}^{\mathrm{T}})^{\mathrm{T}}$.

Consider the congruence system $\mathcal{C} = \mathcal{E} \cup \mathcal{F}$ in $\mathbb{Q}^n$, where $\mathcal{E}$ is a set of equalities and $\mathcal{F}$ is a set of proper congruences. Then the *homogeneous form* for $\mathcal{C}$ is the congruence system $\hat{\mathcal{C}} = \hat{\mathcal{E}} \cup \hat{\mathcal{F}}$ in $\mathbb{Q}^{n+1}$ defined by:

$$\hat{\mathcal{E}} := \Big\{ \big\langle (-b, \boldsymbol{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\boldsymbol{x}} \big\rangle = 0 \;\Big|\; \big(\langle \boldsymbol{a}, \boldsymbol{x}\rangle = b\big) \in \mathcal{E} \Big\},$$
$$\hat{\mathcal{F}} := \Big\{ \big\langle f^{-1}(-b, \boldsymbol{a}^{\mathrm{T}})^{\mathrm{T}}, \hat{\boldsymbol{x}} \big\rangle \equiv_1 0 \;\Big|\; \big(\langle \boldsymbol{a}, \boldsymbol{x}\rangle \equiv_f b\big) \in \mathcal{F} \Big\} \cup \big\{ \langle \boldsymbol{e}_0, \hat{\boldsymbol{x}}\rangle \equiv_1 0 \big\}.$$

The congruence $\langle \boldsymbol{e}_0, \hat{\boldsymbol{x}}\rangle \equiv_1 0$ expresses the fact that $1 \equiv_1 0$. By writing $\hat{\mathcal{E}} = (E^{\mathrm{T}}\boldsymbol{x} = \mathbf{0})$ and $\hat{\mathcal{F}} = (F^{\mathrm{T}}\boldsymbol{x} \equiv_1 \mathbf{0})$, where $E, F \subseteq \mathbb{Q}^{n+1}$, it can be seen that the pair (F, E), called the *matrix form* of $\hat{\mathcal{C}}$, is sufficient to determine $\mathcal{C}$.

Consider next a generator system $\mathcal{G} = (L, Q, P)$ in $\mathbb{Q}^n$. Then the *homogeneous form* for $\mathcal{G}$ is the generator system $\hat{\mathcal{G}} := \big(\hat{L}, \hat{Q} \cup \hat{P}, \{\mathbf{0}\}\big)$ in $\mathbb{Q}^{n+1}$ where

$$\hat{L} := \big\{ (0, \boldsymbol{\ell}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \boldsymbol{\ell} \in L \big\},\ \hat{Q} := \big\{ (0, \boldsymbol{q}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \boldsymbol{q} \in Q \big\},\ \hat{P} := \big\{ (1, \boldsymbol{p}^{\mathrm{T}})^{\mathrm{T}} \;\big|\; \boldsymbol{p} \in P \big\}.$$

The original grid $\mathcal{L} = \mathrm{gcon}(\mathcal{C})$ (resp., $\mathcal{L} = \mathrm{ggen}(\mathcal{G})$) can be recovered from the grid $\hat{\mathcal{L}} = \mathrm{gcon}(\hat{\mathcal{C}})$ (resp., $\hat{\mathcal{L}} = \mathrm{ggen}(\hat{\mathcal{G}})$) since $\mathcal{L} = \big\{ \boldsymbol{v} \in \mathbb{R}^n \;\big|\; (1, \boldsymbol{v}^{\mathrm{T}})^{\mathrm{T}} \in \hat{\mathcal{L}} \big\}$. Note that, if $(\mathcal{C}, \mathcal{G})$ is a double description for a grid and $\hat{\mathcal{C}}$ and $\hat{\mathcal{G}}$ are homogeneous forms for $\mathcal{C}$ and $\mathcal{G}$, then $(\hat{\mathcal{C}}, \hat{\mathcal{G}})$ is also a double description.

Converting Representations. By considering the matrix forms of the (homogeneous forms of the) representations, we can build the conversion algorithms on top of those for matrix inversion. For an informal explanation why this is appropriate, suppose that the generator system $\mathcal{G} = \big(\varnothing, Q, \{\mathbf{0}\}\big)$ in $\mathbb{Q}^n$ is in minimal form and Q is a non-singular square matrix. Letting $\mathcal{L} = \mathrm{ggen}(\mathcal{G}) = \{\, Q\boldsymbol{\pi} \mid$

$\boldsymbol{\pi} \in \mathbb{Z}^n \}$, then we also have $\mathcal{L} = \{ \boldsymbol{v} \in \mathbb{R}^n \mid Q^{-1}\boldsymbol{v} \equiv_1 0 \}$, so that $(Q^{-1}, \varnothing)$ is the matrix form of a congruence system for the same grid $\mathcal{L}$. Similarly we can use matrix inversion to convert the matrix form of a homogeneous congruence system in minimal form consisting of n proper congruences for a grid $\mathcal{L}$ to a generator system for $\mathcal{L}$. When the matrices to be inverted have less than n linearly independent columns, the algorithms first add vectors $\boldsymbol{e}_i$ where $1 \leq i \leq n$, as necessary, so as to make the matrices non-singular and hence invertible.

Proposition 6. *Let $\mathcal{C}$ be a congruence system in $\mathbb{Q}^n$ in minimal form; (F, E) the matrix form of the homogeneous form for $\mathcal{C}$; N a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\boldsymbol{e}_i$, $i \in \{0, \ldots, n\}$, and such that $(N, \hat{F}, \hat{E})$ is square and nonsingular; and $(\hat{L}, \hat{Q}, M) := \big((N, \hat{F}, \hat{E})^{-1}\big)^{\mathrm{T}}$ where $\#\,\hat{L} = \#\,N$, $\#\,\hat{Q} = \#\,\hat{F}$ and $\#\,M = \#\,\hat{E}$. Then $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ is the homogeneous form for a generator system $\mathcal{G}$ in minimal form and* $\mathrm{ggen}(\mathcal{G}) = \mathrm{gcon}(\mathcal{C})$.

Proposition 7. *Let $\mathcal{G}$ be a generator system in $\mathbb{Q}^n$ in minimal form; $\hat{\mathcal{G}} = \big(\hat{L}, \hat{Q}, \{\mathbf{0}\}\big)$ the homogeneous form for $\mathcal{G}$; M a matrix in $\mathbb{Z}^{n+1}$ whose vectors are of the form $\boldsymbol{e}_i$, $i \in \{0, \ldots, n\}$, and such that $(\hat{L}, \hat{Q}, M)$ is square and nonsingular; and $(N, \hat{F}, \hat{E}) := \big((\hat{L}, \hat{Q}, M)^{-1}\big)^{\mathrm{T}}$ where $\#\,N = \#\,\hat{L}$, $\#\,\hat{F} = \#\,\hat{Q}$ and $\#\,\hat{E} = \#\,M$. Then $(\hat{F}, \hat{E})$ is the matrix form of the homogeneous form for a congruence system $\mathcal{C}$ in minimal form and* $\mathrm{gcon}(\mathcal{C}) = \mathrm{ggen}(\mathcal{G})$.

Both algorithms just perform matrix inversion; so their complexity depends on the inversion algorithm adopted in the implementation. As far as we know, the current best theoretical worst-case complexity is $\mathrm{O}\big(n^{2.376}\big)$ [5]. Note that, in the current implementation in the PPL, the conversion algorithm is based on the Gaussian elimination method, which has complexity $\mathrm{O}\big(n^3\big)$.

5 Grid Widening

A simple and general characterization of a widening for enforcing and accelerating convergence of an upward iteration sequence is given in [6–9]. We assume here a minor variation of this classical definition (see footnote 6 in [9, p. 275]).

Definition 5. (Widening.) *Let $\langle D, \vdash, \mathbf{0}, \oplus \rangle$ be a join-semilattice. The partial operator $\nabla\colon D \times D \rightarrowtail D$ is a* widening *if*

1. *for each $d_1, d_2 \in D$, $d_1 \vdash d_2$ implies that $d_1 \nabla d_2$ is defined and $d_2 \vdash d_1 \nabla d_2$;*
2. *for each increasing chain $d_0 \vdash d_1 \vdash \cdots$, the increasing chain defined by $d'_0 := d_0$ and $d'_{i+1} := d'_i \nabla (d'_i \oplus d_{i+1})$, for $i \in \mathbb{N}$, is not strictly increasing.*

In addition to the formal requirements in Definition 5, it is also important to have a widening that has an efficient implementation, preferably, one that depends on a simple syntactic mapping of the representations. At the same time, so that the widening is well-defined, the result of this operation should be independent of the actual representation used. For this reason, the two widenings we propose assume specific minimal forms for the congruence and generator systems.

Definition 6. *A congruence system $\mathcal{C}$ is in* strong minimal form *if, for each pair of distinct proper congruences, $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \equiv_1 b$ and $\langle \boldsymbol{c}, \boldsymbol{x} \rangle \equiv_1 d$ in $\mathcal{C}$, if $\operatorname{piv}_<(\boldsymbol{c}) = k > 0$, then $-c_k < 2a_k \leq c_k$. A generator system $\mathcal{G} = \big((L, Q, P)\big)$ in $\mathbb{Q}^n$ is in* strong minimal form *if $\mathcal{G}$ is in minimal form and, for each pair of distinct parameters $\boldsymbol{u}, \boldsymbol{v} \in Q$, if $\operatorname{piv}_>(\boldsymbol{v}) = k \leq n$, then $-v_k < 2u_k \leq v_k$.*

Proposition 8. *There exists an algorithm with complexity $\mathrm{O}\big(n^3\big)$ for converting a congruence system $\mathcal{C}$ (resp., generator system $\mathcal{G}$) in minimal form to a congruence system $\mathcal{C}'$ (resp., generator system $\mathcal{G}'$) in strong minimal form such that $\operatorname{gcon}(\mathcal{C}) = \operatorname{gcon}(\mathcal{C}')$ (resp., $\operatorname{ggen}(\mathcal{G}) = \operatorname{ggen}(\mathcal{G}')$).*

The widenings defined below use either the congruence or the generator systems.

Definition 7. *Let $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2)$ be two grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{C}_1$ is in minimal form and $\mathcal{C}_2$ is in strong minimal form. Then the* grid widening $\mathcal{L}_1 \nabla_{\mathrm{c}} \mathcal{L}_2$ *is defined by*

$$\mathcal{L}_1 \nabla_{\mathrm{c}} \mathcal{L}_2 := \begin{cases} \mathcal{L}_2, & \text{if } \mathcal{L}_1 = \varnothing \text{ or } \dim(\mathcal{L}_1) < \dim(\mathcal{L}_2), \\ \operatorname{gcon}(\mathcal{C}_S), & \text{otherwise,} \end{cases}$$

where $\mathcal{C}_S := \{\, \gamma \in \mathcal{C}_2 \mid \exists \beta \in \mathcal{C}_1 \,.\, \beta \Uparrow \gamma \,\}$.

Definition 8. *Let $\mathcal{L}_1 = \operatorname{ggen}(\mathcal{G}_1)$ and $\mathcal{L}_2 = \operatorname{ggen}(\mathcal{G}_2)$ be two grids in $\mathbb{G}_n$ such that $\mathcal{L}_1 \subseteq \mathcal{L}_2$, $\mathcal{G}_1 = (L_1, Q_1, P_1)$ is in minimal form and $\mathcal{G}_2 = (L_2, Q_2, P_2)$ is in strong minimal form. Then the* grid widening $\mathcal{L}_1 \nabla_{\mathrm{g}} \mathcal{L}_2$ *is defined by*

$$\mathcal{L}_1 \nabla_{\mathrm{g}} \mathcal{L}_2 := \begin{cases} \mathcal{L}_2, & \text{if } \mathcal{L}_1 = \varnothing \text{ or } \dim(\mathcal{L}_1) < \dim(\mathcal{L}_2); \\ \operatorname{ggen}(\mathcal{G}_S), & \text{otherwise,} \end{cases}$$

where $\mathcal{G}_S := \big(L_2 \cup (Q_2 \setminus Q_S), Q_S, P_2\big)$ and $Q_S := \{\, \boldsymbol{v} \in Q_2 \mid \exists \boldsymbol{u} \in Q_1 \,.\, \boldsymbol{u} \Downarrow \boldsymbol{v} \,\}$.

Proposition 9. *The operators '∇_{c}' and '∇_{g}' are both widenings on $\mathbb{G}_n$.*

In Definition 7, it is required that $\mathcal{C}_2$ is in strong minimal form. The following example shows that this is necessary for the operator '∇_{c}' to be well-defined.

Example 4. Let $\mathcal{L}_1 := \operatorname{gcon}(\mathcal{C}_1)$, $\mathcal{L}_2 := \operatorname{gcon}(\mathcal{C}_2)$ and $\mathcal{L}'_2 := \operatorname{gcon}(\mathcal{C}'_2)$ where $\mathcal{C}_1 = \{x \equiv_2 0,\ y \equiv_2 0\}$, $\mathcal{C}_2 = \{x \equiv_1 0,\ x + y \equiv_2 0\}$, $\mathcal{C}'_2 = \{x \equiv_1 0,\ 3x + y \equiv_2 0\}$; then $\mathcal{L}_2 = \mathcal{L}'_2$. Note that only $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form. Therefore, assuming $\mathcal{C}_S$ (resp., ${\mathcal{C}_S}'$) is defined as in Definition 7 using $\mathcal{C}_1$ and $\mathcal{C}_2$ (resp., $\mathcal{C}_1$ and $\mathcal{C}'_2$), we have $\mathcal{C}_S = \{x + y \equiv_2 0\}$ and ${\mathcal{C}_S}' = \{3x + y \equiv_2 0\}$. Thus $\mathcal{L}_1 \nabla_{\mathrm{c}} \mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_S) \neq \operatorname{gcon}({\mathcal{C}_S}')$.

Example 5. To see that the widenings depend on the variable ordering, consider the grids $\mathcal{L}_1 = \operatorname{gcon}(\mathcal{C}_1) = \operatorname{gcon}(\mathcal{C}'_1)$ and $\mathcal{L}_2 = \operatorname{gcon}(\mathcal{C}_2) = \operatorname{gcon}(\mathcal{C}'_2)$ in $\mathbb{G}_2$, where

$$\begin{aligned} \mathcal{C}_1 &:= \{5x + y \equiv_1 0,\ 22x \equiv_1 0\}, \quad \mathcal{C}_2 := \{5x + y \equiv_1 0,\ 44x \equiv_1 0\}, \\ \mathcal{C}'_1 &:= \{9y + x \equiv_1 0,\ 22y \equiv_1 0\}, \quad \mathcal{C}'_2 := \{9y + x \equiv_1 0,\ 44y \equiv_1 0\}. \end{aligned}$$

Assume for $\mathcal{C}_1$ and $\mathcal{C}_2$ that the variables are ordered so that x precedes y, as in the vector $(x, y)^{\mathrm{T}}$; then, $\mathcal{C}_1$ and $\mathcal{C}_2$ are in strong minimal form and, according to Definition 7, we obtain $\mathcal{L}_1 \nabla_{\mathrm{c}} \mathcal{L}_2 = \mathrm{gcon}\big(\{5x + y \equiv_1 0\}\big)$. On the other hand, $\mathcal{C}'_1$ and $\mathcal{C}'_2$ are in strong minimal form when taking the variable order where y precedes x. In this case, by Definition 7, $\mathcal{L}_1 \nabla_{\mathrm{c}} \mathcal{L}_2 = \mathrm{gcon}\big(\{9y + x \equiv_1 0\}\big)$.

6 Conclusion

We have defined a domain of *grids* and shown that any element may be represented either by a congruence system which is a finite set of congruences (either equalities or proper congruences); or a generator system which is a triple of finite sets of vectors (denoting sets of lines, parameters and points). Assuming such a system in $\mathbb{Q}^n$ has m congruences or generators, then the minimization algorithms have worst-case complexity $\mathrm{O}\big(n^2 m\big)$. It is shown that any matrix inversion algorithms such as Gaussian elimination which has complexity $\mathrm{O}\big(n^3\big)$, can be used for converting between generator and congruence systems in minimal form. Thus, the complexity of converting any system with m elements is no worse than $\mathrm{O}\big(n^2 m\big)$ if $m > n$ and $\mathrm{O}\big(n^3\big)$, otherwise.

The minimization and conversion algorithms, form the basis for a double description method for grids so that any generator or congruence systems, possibly in minimal form, can be provided on demand; the complexity of such a provision being as stated above. Assuming this method, we have shown that operations for comparison, intersection and grid join are straightforward. The complexity of comparing two grids is $\mathrm{O}\big(n^3\big)$ but, for just checking equality when it is already known that one of the grids is a subset of the other, we have described simpler procedures with complexity $\mathrm{O}\big(n\big)$. The intersection and grid join just take the union of the congruence or generator systems, respectively, so that, from a theoretical perspective, these have complexity $\mathrm{O}\big(n\big)$. However, in the implementation, we assume a common divisor for all the coordinates or coefficients in the system; hence, combining the systems requires changing the denominators of both components to their least common multiple with a consequential need to scale all the numerators in the representation; giving a worst-case complexity of $\mathrm{O}\big(n^2\big)$. We have also described an algorithm for computing the grid difference with complexity $\mathrm{O}\big(n^4\big)$. Observe that this operator is useful in the specification of the certificate-based widening for the grid powerset domain [3].

The grid domain is implemented in the PPL [2, 4] following the approach described in this paper. Among the tests available in the PPL are the examples in this paper and implementations of the running examples in [22, 23]. The PPL provides full support for lifting any domain to the powerset of that domain, so that a user of the PPL can experiment with powersets of grids and the extra precision this provides. An interesting line of research is the combination of the grids domain with the polyhedral domains provided by the PPL: not only the $\mathbb{Z}$-polyhedra domain, but also many variations such as the grid-polyhedra, grid-octagon, grid-bounded-difference, grid-interval domains and their powersets.

References

1. C. Ancourt. *Génération Automatique de Codes de Transfert pour Multiprocesseurs à Mémoires Locales*. PhD thesis, Université de Paris VI, March 1991.
2. R. Bagnara, P. M. Hill, and E. Zaffanella. *The Parma Polyhedra Library User's Manual*. Department of Mathematics, University of Parma, Parma, Italy, release 0.9 edition, March 2006. Available at `http://www.cs.unipr.it/ppl/`.
3. R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. *Software Tools for Technology Transfer*, 8(4/5):449–466, 2006.
4. R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 213–229, Madrid, Spain, 2002. Springer-Verlag, Berlin.
5. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.
6. P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In B. Robinet, editor, *Proceedings of the Second International Symposium on Programming*, pages 106–130, Paris, France, 1976. Dunod, Paris, France.
7. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252, New York, 1977. ACM Press.
8. P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, 1992.
9. P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming*, volume 631 of *Lecture Notes in Computer Science*, pages 269–295, Leuven, Belgium, 1992. Springer-Verlag, Berlin.
10. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 84–96, Tucson, Arizona, 1978. ACM Press.
11. R. Giacobazzi, editor. *Static Analysis: Proceedings of the 11th International Symposium*, volume 3148 of *Lecture Notes in Computer Science*, Verona, Italy, 2004. Springer-Verlag, Berlin.
12. P. Granger. Static analysis of arithmetical congruences. *International Journal of Computer Mathematics*, 30:165–190, 1989.
13. P. Granger. *Analyses Sémantiques de Congruence*. PhD thesis, École Polytechnique, 921128 Palaiseau, France, July 1991.
14. P. Granger. Static analysis of linear congruence equalities among variables of a program. In Samson Abramsky and T. S. E. Maibaum, editors, *TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Volume 1: Colloquium on Trees in Algebra and Programming (CAAP'91)*, volume 493 of *Lecture Notes in Computer Science*, pages 169–192, Brighton, UK, 1991. Springer-Verlag, Berlin.
15. P. Granger. Static analyses of congruence properties on rational numbers (extended abstract). In P. Van Hentenryck, editor, *Static Analysis: Proceedings of the*

4th International Symposium, volume 1302 of *Lecture Notes in Computer Science*, pages 278–292, Paris, France, 1997. Springer-Verlag, Berlin.
16. M. Karr. Affine relationships among variables of a program. *Acta Informatica*, 6:133–151, 1976.
17. S. Larsen, E. Witchel, and S. P. Amarasinghe. Increasing and detecting memory address congruence. In *Proceedings of the 2002 International Conference on Parallel Architectures and Compilation Techniques (PACT'02)*, pages 18–29, Charlottesville, VA, USA, 2002. IEEE Computer Society Press.
18. V. Loechner. *PolyLib*: A library for manipulating parameterized polyhedra. Available at `http://icps.u-strasbg.fr/~loechner/polylib/`, March 1999. Declares itself to be a continuation of [30].
19. A. Miné. A few graph-based relational numerical abstract domains. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 117–132, Madrid, Spain, 2002. Springer-Verlag, Berlin.
20. T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games – Volume II*, number 28 in Annals of Mathematics Studies, pages 51–73. Princeton University Press, Princeton, New Jersey, 1953.
21. M. Müller-Olm and H. Seidl. Precise interprocedural analysis through linear algebra. In N. D. Jones and X. Leroy, editors, *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2004)*, pages 330–341, Venice, Italy, 2004. ACM Press.
22. M. Müller-Olm and H. Seidl. Analysis of modular arithmetic. In M. Sagiv, editor, *Programming Languages and Systems, Proceedings of the 14th European Symposium on Programming*, volume 3444 of *Lecture Notes in Computer Science*, pages 46–60, Edinburgh, UK, 2005. Springer-Verlag, Berlin.
23. M. Müller-Olm and H. Seidl. A generic framework for interprocedural analysis of numerical properties. In C. Hankin and I. Siveroni, editors, *Static Analysis: Proceedings of the 12th International Symposium*, volume 3672 of *Lecture Notes in Computer Science*, pages 235–250, London, UK, 2005. Springer-Verlag, Berlin.
24. S. P. K. Nookala and T. Risset. A library for Z-polyhedral operations. *Publication interne* 1330, IRISA, Campus de Beaulieu, Rennes, France, 2000.
25. P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra. Technical Report 1016, IRISA, Campus Universitaire de Bealieu, Rennes, France, July 1996.
26. P. Quinton, S. Rajopadhye, and T. Risset. On manipulating Z-polyhedra using a canonic representation. *Parallel Processing Letters*, 7(2):181–194, 1997.
27. E. Rodríguez-Carbonell and D. Kapur. An abstract interpretation approach for automatic generation of polynomial invariants. In Giacobazzi [11], pages 280–295.
28. S. Sankaranarayanan, H. Sipma, and Z. Manna. Constraint-based linear-relations analysis. In Giacobazzi [11], pages 53–68.
29. A. Schrijver. *Theory of Linear and Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1999.
30. D. K. Wilde. A library for doing polyhedral operations. Master's thesis, Oregon State University, Corvallis, Oregon, December 1993. Also published as IRISA *Publication interne* 785, Rennes, France, 1993.